Institute for Scientific Computing Research

ISCR Summer Student Program

ISCR Summer Student Program

Each summer the ISCR runs an extensive summer program for students ranging from those still in high school to those completing their PhDs. Students are assigned specific projects appropriate to their research capabilities. A technical mentor oversees the work and provides guidance. While in residence, students attend research seminars, prepare a poster summary of their work, and write up a progress report on their efforts.

Time-Critical Occlusion Culling: A Multiresolution Approach

Lucas Ackerman
Worcester Polytechnic Institute
Mentor
Mark Duchaineau
CASC

Summary

This research is the result of an ongoing three-year collaboration with Mark Duchaineau on scalable visibility algorithms.

We have developed an algorithm for conservative visibility determination on multiresolution surfaces, suitable for complementing continuous level-of-detail (LOD) rendering algorithms. Fundamental issues include the feedback between LOD and visibility requirements, the methods of performing occlusion with elements of multiresolution geometry, and the unique benefits of calculating visibility in an inherently multiresolution domain. This is the only approach known to be adaptive in object space on both occludee and occluder hierarchies. It requires no precomputation beyond the nested error bounds used for continuous LOD display (as in the ROAM algorithm) and is amenable to being made progressive and output-sensitive with respect to the computed visible set.

This algorithm complements the ROAM family of continuous LOD algorithms.

Future plans include alternate implementations to improve performance, address shortcomings, and increase flexibility, and a formal paper for publication.

Multigrid for Maxwell's Equations on Structured and Unstructured Grids

David Alber
University of Illinois, Urbana–Champaign

Mentor

Jim Jones
CASC

Summary

Maxwell's equations unify the concepts of the electric and magnetic fields. As such, they are fundamental to electromagnetics and constitute an important problem to discretize and solve numerically with good accuracy and speed. Unfortunately, the discrete Maxwell's problem is not solved well by standard methods because of the operator of the discretized problem. The Maxwell operator has components that are near null-space but are not smooth. This means that standard relaxation will not damp this error component because it is algebraically smooth, and coarse grid correction will not remove the error because it cannot be represented on a coarse grid. This difficulty suggests that new smoothers, new grid-transfer operators, or both need to be developed.

Both the definite and indefinite formulations of Maxwell's equations were examined this summer. In the definite formulation, the near null-space components of the operator were handled by an overlapping Schwarz smoother, as suggested by Arnold, Falk, and Winther. Good rates for this problem on a structured grid were achieved using this smoother and standard coarsening on the unknowns. A solver for the definite Maxwell system on unstructured grids was also developed. This solver uses element agglomeration techniques to select the coarse grid and also uses the overlapping Schwarz smoother. The rates that have been observed for this solver are as good as the rates from the solver on the structured grid. However, it is premature to draw too many conclusions from this result because of limited testing. More work needs to be done on this solver. The indefinite formulation of Maxwell's equations offers additional complications in the form of near null-space plane waves. These waves are difficult to eliminate, and thus far, a good solver has not been found for this problem.

This project will continue with further work on solvers for both definite Maxwell's on unstructured grids and the indefinite Maxwell's formulation. Many pieces of the solver on the unstructured grid will be examined, including the agglomeration routines, the smoother, and the prolongation operator.

Software Quality Assurance and Miscellaneous Changes in DRACO

John C. Anderson
University of the Pacific
Mentor
Benjamin T. Grover
DCOM

Summary

In large software projects, the ability to quantitatively assess both correctness and quality is important. It is impractical to employ traditional quality control techniques such as peer-review when dealing with a large, rapidly changing code base. DRACO, a parallel mesh generation tool with over 275,000 lines of code, was a prime candidate for two modern methods of quality assurance: Design By Contract (DBC) programming and unit testing [2]. The goal of the investigation was to implement software quality tools for DRACO addressing the need for DBC as well as unit testing.

Over the summer, I was able to implement a complete set of tools for software quality assurance in DRACO. These tools include two DBC systems, one for C++ and another for Python, an extended testing framework complete with over 175 unit tests and more testing modes, and a robust C++ debug flag system with an easy-to-use Python interface.

After finishing the above software quality enhancements, I worked to improve and extend other areas of DRACO. Improvements were made to the build system, installation script, the GUI, and the handling of material attributes on meshes. Extensions included the development of a function browser for executing DRACO commands from the GUI, a Boolean Operations dialog, and the addition of a flexible view mode system for DRACO's graphical display.

Performance of I/O Nodes in the BlueGene/L Supercomputer

Scott Banachowski
University of California, Santa Cruz
Mentor
Kim Yates
CAR

Summary

The BlueGene/L supercomputer (BG/L) will be a massively parallel system of 65,536 compute nodes. The system software of BG/L is architecturally organized into 1024 logical groups of 64 compute processors. To offload many operating system tasks such as file I/O, sockets, and debugger services from compute nodes, each logical group employs an additional I/O node that runs an embedded version of Linux, whereas compute nodes run a small, simple, fast kernel. The compute nodes ship I/O requests, such as file reads and writes and other system operations, to their I/O node for execution. The goal of the project is to investigate if an I/O node has the computational power and resources to support 64 compute nodes without performance bottlenecks.

A suite of benchmarking software was developed for this project. It consists of two applications: a parallel application, which runs on the compute nodes and creates load to stress the I/O node by making bulk file operations, and a monitor application, which runs on the I/O node and measures statistics about the node's resource usage, including memory, network bandwidth, and processor idleness. The system software that runs on the I/O node was not available to view or modify, so we must determine what is happening in the I/O node through other means. The I/O node runs a version of Linux and, unlike the compute nodes, is capable of multitasking. Therefore, it is possible to launch an application that makes resource measurements on the I/O node during the execution of the load experiments. The I/O node monitoring application uses facilities that Linux provides for system status through the proc file system (a pseudo-file system that supports a filelike interface to kernel data structures). During the summer, the benchmarking applications were developed and tested on the BG/L simulator. The simulator was useful for application debugging, but because it is not timing-accurate, any performance inferred from it is not representative of actual performance. In the final week of the practicum, the tests were executed on a 128-node BG/L prototype at IBM Watson Research Center. Although many of the features we wished to test were not entirely implemented, we were able to capture some data from real hardware. Our data shows that the I/O nodes appear capable of scaling to the task of handling the system requests of 64 compute nodes.

Because the BG/L system is still in development, it is a bit premature to measure its performance. For example, the file system prototype is not yet performing near the specified level, so it created an artificial bottleneck in the measured performance. In the future, when the system is more mature, the software developed in this project may be used to further study the interaction between BG/L compute and I/O nodes.

Continued

Scott Banachowski
University of California, Santa Cruz

Summary continued

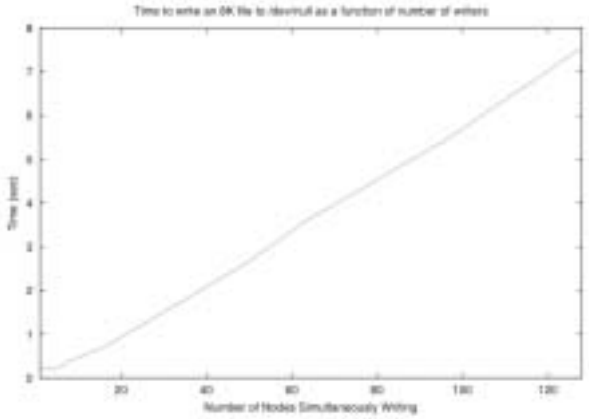


Fig. 1: We varied the number of nodes concurrently writing data to determine how long processing takes when each node writes 8 Kb of file data. To remove the effect of file system latency and measure only operating system overhead, all the data are directed to a null file.

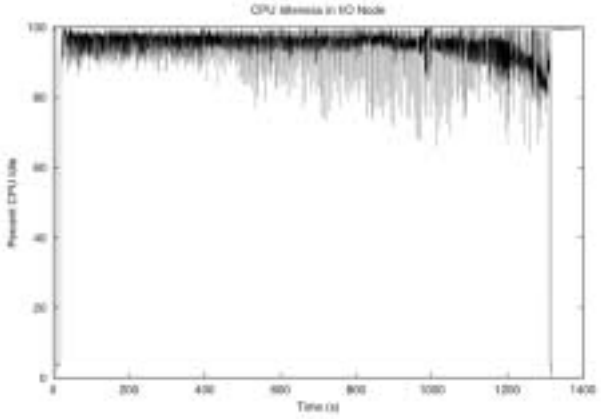


Fig. 2: This is the amount of user and system CPU remaining idle when 64 nodes simultaneously write a file.

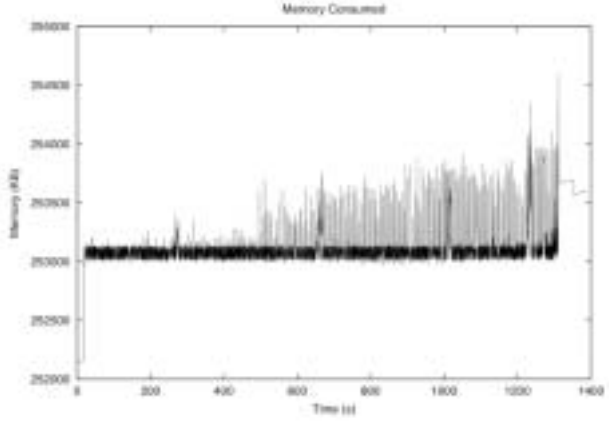


Fig. 3: This plot shows the memory consumed by the I/O node when 64 nodes write a file.

Slow-Growing Subdivision for Generic Unstructured Meshes

Janine Bennett
University of California, Davis
Mentor
Valerio Pascucci
CASC

Summary

Scientific visualization techniques strive to maintain stringent error bounds to ensure that the visual feedback provided to the user or scientist is accurate and reliable. Scientific simulations often use unstructured mesh domains of arbitrary topology with data values sampled at the mesh nodes. However, a hierarchical, regular mesh structure is preferred for efficient, multiresolution access, computation, analysis, and rendering of the data. As a result, the original meshes are often modified for visualization or other postprocessing purposes using interpolation schemes that differ from those used by the simulation. The resulting output is inherently inaccurate, demonstrating the fundamental inefficiency of current representation schemes used for unstructured meshes.

Subdivision methods are one of the most successfully used techniques in multiresolution data representation for surface meshes. Initial research has been done to extend these techniques to the multiresolution representation of volumetric data. For example, the slow-growing subdivision (SGS), first described by Pascucci, is a subdivision method that generalizes to volumetric and higher dimensional meshes. We started with a basic implementation of this approach for convex domains and regular curvilinear grids, where a mapping to a regular structure is provided explicitly (see figure). In this initial stage, we have successfully provided new wavelet representations that can be used to represent associated field data.

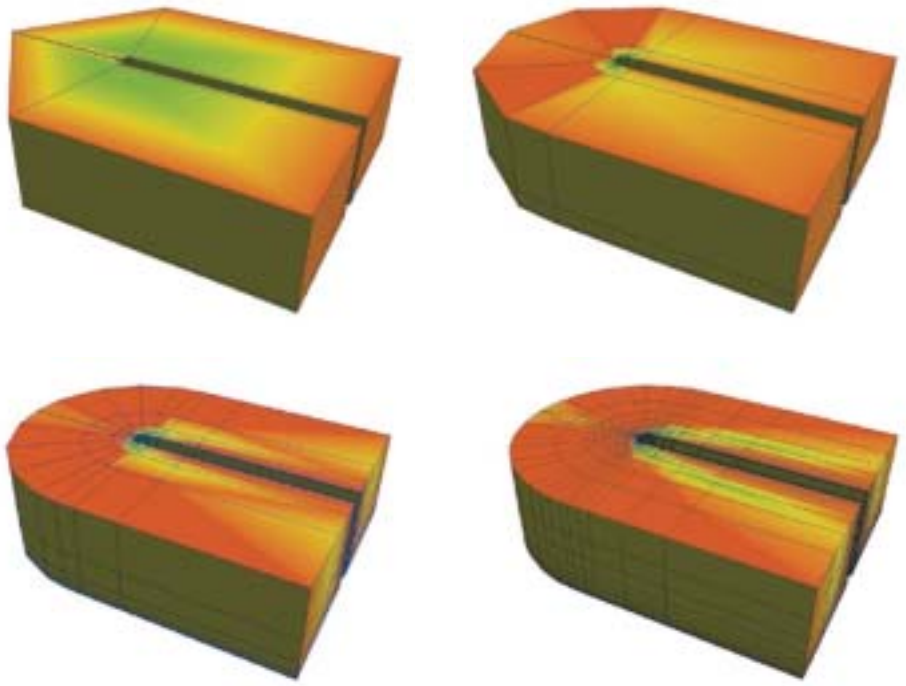
Our main focus for the summer was to complete the postprocessing, compression, and visualization process for SGS representation of arbitrary unstructured meshes. To accomplish this, we decompose the input domain into regions with convexlike connectivity. To this end, we are experimenting with a region growing technique that starts from single-cell sets that are expanded as long as their structural properties are maintained. After this process, we map each region to a convex domain to obtain the necessary isomorphism from unstructured mesh connectivity to subdivision connectivity.

While the completion of a method to remesh a generic unstructured mesh to a mesh on which SGS could be performed is a high priority and is important for visualization purposes, our long-term goal is to provide a unified representation of unstructured mesh data that can be used in simulations and visualization techniques. This data representation would not only produce final images that are inherently more accurate, but could also be used to make simulations more efficient. To this end, we need to work on two major challenges: analyze the formal asymptotic properties and the practical behavior of the nested function spaces associated with the subdivision process and determine how to maintain mesh quality metrics while performing the recursive subdivision process.

Continued

Janine Bennett
University of California Davis

Summary continued



SGS refinement of a nonrectilinear mesh (blunt fin from NASA) with temperature field visualized by pseudocolor map.

Network Vulnerability Assessment (NVA) Project

Bridget Benson
California Polytechnic State University

Mentor

Terry Brugger
NAIC

Summary

The goal of the network vulnerability assessment (NVA) project is to allow computer security personnel to easily perform a network vulnerability assessment. The NVA project involves developing and using tools to collect as much data as possible about a particular network and then displaying this data in a graphical user interface (GUI) to allow computer security personnel and penetration testers to understand the network and look for vulnerabilities.

The NVA project's GUI, known as Graph Viewer, needed bug fixes and enhancements to make it a more useful tool for finding vulnerabilities in the network. Some of these enhancements involved adding color keys and extending the viewer's capability with grouping nodes in the network.

To make enhancements to Graph Viewer, I had to become familiar with its large, existing Java code base. I used the integrated development environment known as Eclipse to step through thousands of lines of code to understand how the viewer was getting, drawing, and arranging the network data.

With the help of my mentor, Terry Brugger, and a new colleague, Chris Brand, with experience in Java Swing and Graphics classes, I was able to make many improvements to Graph Viewer including:

- adding the ability to color network connections based on some information about that connection,
- adding color keys for all color coding done in the viewer,
- fixing a bug to make sure layouts involving groups of computers worked correctly,
- coloring groups of machines,
- extending the expand and reduce functions,
- supporting multiple routes to hosts,
- making labeling by hostname more useful,
- changing the icon of a node in the network based on what server it represents, and
- adding support for analyzing network topology over time.

Through making improvements to Graph Viewer, I acquired a better understanding of the types of vulnerabilities a computer analyst might look for in a network and a great deal of debugging and coding (especially with Java Swing) experience.

The improvements made to the Graph Viewer will aid in meeting the overall objectives of the NVA project. Computer security professionals will continue to make enhancements on Graph Viewer and all other parts of the NVA project every day. With the continual improvements to the NVA project, computer security professionals at Lawrence Livermore National Laboratory are approaching their vision of an automated, comprehensive network vulnerability assessment system.

Massive Volumetric Visualization through Progressive Algorithms and Data Structures

Rita Stefania Borgo
Universita' di Pisa
Mentor
Valerio Pascucci
CASC

Summary

Projects dealing with massive amounts of data need to carefully consider all aspects of data acquisition, storage, retrieval, and navigation. The recent growth in size of large simulation datasets still surpasses the combined advances in hardware infrastructure and processing algorithms for scientific visualization. The cost of storing and visualizing such datasets is prohibitive, so that only one out of every hundred time-steps can be stored and visualized. As a consequence, interactive visualization of result is going to become increasingly difficult, especially as a daily routine from a desktop. High frequency of I/O operations starts dominating the overall running time. In this panorama, the efficiency of a visualization algorithm must be evaluated in the context of end-to-end systems instead of being optimized individually.

Starting from the point that relying on memory layout to speed up the access time is no longer possible because datasets are too large to be kept in main memory or stored on a local disk, a need arises at system level to design the visualization process as a pipeline of modules able to process data in stages, thus creating a flow of data that need to be optimized globally with respect to magnitude and location of available resources.

My objective during the time spent at Lawrence Livermore National Laboratory has been to develop a new progressive visualization algorithm based on edge-bisection refinement, a technique widely used in the meshing community.

The first part of my project was to implement and apply the edge-bisection technique in a 3D environment, with improving visualization of 3D datasets as the overarching goal. Edge-bisection has been implemented based on a set of simple rules that characterize consistently the decomposition of a grid in simplices together with the recursive refinement of the derived simplicial mesh. The result is a new naming scheme that allows representation of an adaptive simplicial mesh with a very low memory footprint.

One property of the subdivision scheme is that it holds implicitly a hierarchical organization of the dataset itself. It has therefore been possible to traverse and organize the input grid in a hierarchical structure (from coarse level to fine level) and, as a consequence, to extract subsequent level of detail for improving display of the output image. As a first step, I have uncoupled data extraction from its display, splitting the overall process between two main threads, one that traverses the input 3D mesh and builds up the hierarchy, and a second that performs traversal of the hierarchy and display of the extracted level of isosurface. This approach allows rendering at any given time partial results while the computation of the complete hierarchy makes progress. The regularity of the hierarchy allows the creation of a good data-partitioning scheme that allows balancing processing time and data migration time. Some results of the progressive behavior of the algorithm are shown in Figure 1.

Continued

Rita Stefania Borgo
Universita' di Pisa

Summary continued

Results in this field are applicable in parallel and distributed computing ranging from a cluster of PCs to more complex and expensive architectures. My next step will be to extend the framework built so far to a shared and distributed memory environment.

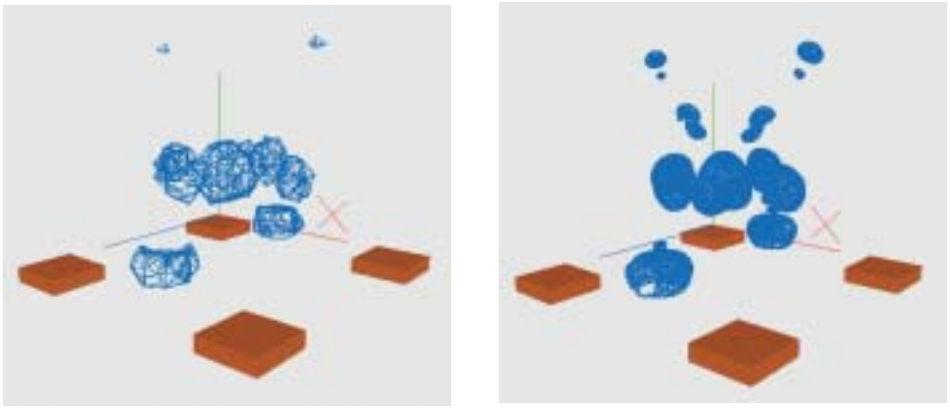


Fig. 1. Illustration of stages of the visualization hierarchy.

Compatible Relaxation in Generalized Algebraic Multigrid (AMG)

James Brannick
University of Colorado, Boulder
Mentor
Rob Falgout
CASC

Summary

In generalizing the algebraic multigrid (AMG) framework, Rob Falgout and Panayot Vassilevski prove that compatible relaxation (CR) can be used to measure the ability of the coarse grid to represent algebraically smooth error. In particular they show that a sufficient condition for guaranteeing the existence of P, the prolongation operator, is that CR is fast to converge. However, the question of how to form such a coarse variable set remains open. Our research focuses on developing a coarsening algorithm that uses CR to construct an adequate coarse variable set.

We consider testing the CR algorithm as outlined in the generalized AMG framework. We first attempted to implement the algorithm exactly as outlined by Falgout and Vassilevski. The results proved to be sensitive to parameters. After trying several different approaches we found a variant of the algorithm that is both robust and efficient in serial.

The results we obtain with this version are comparable with those of Oren Livne. The major drawback of this approach is that the smoother we use is multiplicative, which would be difficult to parallelize and thus not useful in practice, especially in the Hypre AMG package.

Our future work will focus on developing a version of CR that is robust and efficient and uses an additive fine point smoother. The primary issue to address is what to use as a “slowness” measure when choosing coarse points.

Discovery Center Display

Jedidiah Chow

Granada High School, Livermore, California

Mentor

Jean Shuler

ICCD

Summary

The goal of my project was to produce an interactive program that visitors to Lawrence Livermore National Laboratory could use to become informed about the computing goals and accomplishments of the Laboratory's Computation Directorate.

To simplify the project, I used only well-known programs. MicroSoft PowerPoint is the basis for the display program, and it proved adequate for navigating through various topics as well as produced aesthetically pleasing results. Basically, the program provides details about the history of supercomputing at Livermore, visual display equipment available at the Laboratory, and recent uses of the supercomputers there. The project was completed under the supervision of Terry Girill and Bob Howe.

General maintenance and modifications on the program will periodically be necessary (as with any program), and major updates will be accessed as time progresses and the information presented becomes out of date.

Multiresolution Contour Trees in Two Dimensions

Kree Cole-McLaughlin
University of Utah
Mentor
Valerio Pascucci
CASC

Summary

For the past three years, I have been working on algorithms for computing the contour tree and related data structures used in visualization of field data. Simply put, the contour tree is a representation of the connectivity of all of the level sets of a scalar field. More precisely, it is the graph obtained by contracting each connected component of every level set of a scalar field to a point. From the definition, one can see that the nodes of the graph correspond to critical points of the scalar field. Furthermore, it can be shown that if the domain of the field is simply connected, then the constructed graph has no cycles. If the domain has more complex topology, then resulting data structure is called the Reeb graph. Working with Valerio Pascucci of Livermore's Center for Advanced Scientific Computing and other collaborators, we have developed algorithms for efficiently computing the contour tree on domains of any dimension and the Reeb graph on two-dimensional domains.

Lawrence Livermore scientists are interested in analyzing very large datasets from physical simulation codes. These data can have an extremely large number of critical points. However, many of these critical points are often due to noise introduced by the simulation and are topologically less important. The topological noise in the data appears in the contour tree as many edges of the tree that span a very small range of field values. For the contour tree to be useful in the analysis of these datasets, we need some technique for isolating the important topology. In pursuit of this, I worked in the summer of 2003 on an algorithm for computing a multiresolution representation of the contour tree.

Although efficient techniques exist for computing the contour tree we have only been able to visualize a few nontrivial examples. Algorithms for the visualization of very large trees exist; however, they cannot be applied in our case. The problem with visualizing the contour tree is that we want to rank the nodes of the tree by their corresponding critical values. Imposing this condition makes it possible to construct examples of field that have contour trees with nonplanar embeddings. We have also shown that the multiresolution representation of the contour tree solves the problem of visualizing the contour tree.

We have developed an algorithm for computing the multiresolution contour tree for scalar fields in two-dimensional domains. I implemented the algorithm and produced several nontrivial examples using data from a two-dimensional combustion simulation. Using the multiresolution contour tree, we defined an embedding in three-dimensional space that is theoretically free of self intersections. The multiresolution contour tree allows for a visualization algorithm because it defines an order in which the edges of the tree can be drawn. Edges of coarser resolution are drawn, and then the edges of finer resolution are drawn until some threshold is reached.

Fast Implicit Computational Electromagnetics

Dylan Copeland
Texas A&M University
Mentor
Daniel White
CASC

Summary

An important problem for the EMSolve project is solving Maxwell's equations on very fine meshes arising from high-frequency integrated circuit applications. To produce simulations on a long time scale requires implicit time integration. The purpose of the summer assignment was to develop and test a multigrid code for this problem, with the hope that it would be more efficient than the simple conjugate gradient method with diagonal scaling.

In the beginning of the assignment, extensive tests were made with the multilevel code module (ML) software from Sandia National Laboratories. For sufficiently large problems, the Sandia software performed nearly twice as fast as the conjugate gradient method. However, the ML software uses algebraic multigrid, which does not take advantage of the rectilinear meshes associated with the integrated circuit problems in which we are interested.

Therefore, we developed a geometric multigrid code explicitly for rectilinear meshes, using hexahedral Nedelec edge elements. Solving Maxwell's equations implicitly poses some technical problems because of the large null space of the curl operator. To effectively eliminate these problems, we implemented a multigrid smoother developed by R. Hiptmair.

We ran numerical experiments with uniform and nonuniform mesh spacing, as well as multiple materials. In all cases, the geometric multigrid code performed much better than both the conjugate gradient method and explicit time stepping. For the simple case of a uniform mesh with only one material, our code was four times as fast as the conjugate gradient method. For more complex problems with two materials or nonuniform mesh spacing, the geometric multigrid code ran in about the same amount of time as in the uniform case, whereas the conjugate gradient method took more time than we could wait. Thus, our code is very efficient and performs well on practical problems similar to those posed by the integrated circuits we are interested in.

Although we have done some simple experiments with the geometric multigrid solver, it remains to be seen how well it performs on real problems. Ultimately, we would like to compare the performance of our code with that of the Sandia solver on a mesh arising from an integrated circuit. As an application of the code, we would like to apply multigrid as a preconditioner in the computation of eigenvalues for Maxwell's equations by subspace iteration.

Multigrid Preconditioning of Discontinuous Galerkin Methods for Elliptic Problems

Veselin Dobrev
Texas A&M University
Mentor
Panayot Vassilevski
CASC

Summary

The discontinuous Galerkin (DG) methods are relatively new tools for numerical solution of partial differential equations. Their main advantage compared to standard finite-element methods is that they are suitable for approximating solutions with discontinuities (jumps), shocks, or boundary layers. But they give rise to generally more complicated bilinear forms, larger discrete spaces, and linear systems with denser sparse matrices, which in some cases are nonsymmetric even for symmetric continuous problems. All of these factors increase the cost of solving the arising discrete problems, and thus efficient solution algorithms become very important. Multigrid methods are well known to be extremely efficient for preconditioning large linear systems obtained from discretizations of various types of PDEs. Therefore, they are a natural choice for the preconditioning of DG methods as well.

Three DG methods for elliptic problems were considered: the interior penalty (IP) method, the method of Baumann and Oden, and the nonsymmetric interior penalty Galerkin (NIPG) method. They were implemented for general tetrahedral meshes, and the code was parallelized using the Message Passing Interface (MPI) and the Hypre library.

Using the developed code, the convergence rates of the methods were tested numerically with piecewise linear and piecewise quadratic functions. The results for the latter agree with the theoretically known estimates, but for the former, we observe optimal convergence for all three methods, which has been proven only for the IP method.

The performance of two multigrid preconditioners was tested. Both of them use the natural embeddings of the discontinuous finite element spaces to define the interpolation/restriction, which leads to purely local operators. The first preconditioner tested was a variable V-cycle multigrid with diagonal smoother, whose diagonal entries are the row-sums of the absolute values of the elements of the matrix. The second one was V-cycle multigrid in which the ParaSails preconditioner from the Hypre library was used as the smoother. The first observation in these tests was that neither of the two preconditioners worked well for the method of Baumann and Oden, which requires further investigation. With the other two methods (IP and NIPG), both preconditioners performed very well, in that the number of iterations in the iterative method (GMRES for NIPG, PCG for IP) was bounded uniformly for all refinement levels.

Interesting directions for further studies are to understand better why the multigrid preconditioners failed to perform well for the method of Baumann and Oden and then to try to design better smoother and/or interpolation operators. Another interesting problem is to justify theoretically the numerically observed optimal convergence rates for piecewise linear functions. Last, but not least, the parallel scalability of the resulting algorithms should be tested and some possible improvements implemented.

This work was performed in collaboration with Raytcho Lazarov (Texas A&M University) and Panayot Vassilevski (Center for Advanced Scientific Computing, Lawrence Livermore National Laboratory).

Determining the Phenotypic Expression of Cytochrome P4501A2 Using Caffeine as a Probe

Tammi Duncan
Diné College
Mentor
Michael Malfatti
Lawrence Livermore National Laboratory

Summary

Cytochrome P4501A2 (CYP1A2) is involved in the metabolism of many compounds, including 2-amino-1-methyl-6-phenylimidazo[4,5-b]pyridine (PhIP) and caffeine. PhIP is a heterocylic amine that is formed in certain cooked skeletal meats such as beef, chicken, pork, and fish when prepared under common cooking practices and may potentially become carcinogenic to humans when metabolized by CYP1A2. CYP1A2 activity can vary among individuals, which could alter the activation rate of compounds like PhIP. Therefore, to better evaluate the potential risks from PhIP exposure and bioactivation, determining individual expression of CYP1A2 is important.

Earlier studies have shown that caffeine is most commonly used as a probe. Therefore, we used CYP1A2 mediated caffeine metabolism to determine if individuals have a fast or slow CYP1A2 phenotype. We tested the rate of caffeine metabolism using saliva samples from volunteers who were dosed orally with 100 mg of caffeine. The samples were collected at different times (0 hr, 4 hr, 6 hr, 8 hr). First, the samples were collected and prepared for HPLC analysis. Second, the samples were injected into the HPLC to separate the caffeine metabolites. Caffeine and the CYP1A2 mediated caffeine metabolite, paraxanthine (17X), were identified and quantified based on retention time with known standards. Based on our data of paraxanthine to caffeine ratios from saliva samples collected at 6 hrs, generalizations could be made that designate individuals as fast, slow, or intermediate CYP1A2 metabolizers (see Figure 1). Out of 23 individuals tested, 8 were designated as fast, 10 were slow, and 5 were intermediate (see Table 1).

Overall the results of this study have furthered our understanding of carcinogens produced by our body. Future studies may indicate whether individuals with high CYP1A2 activity will be more susceptible to the carcinogenic risks associated with PhIP exposure.

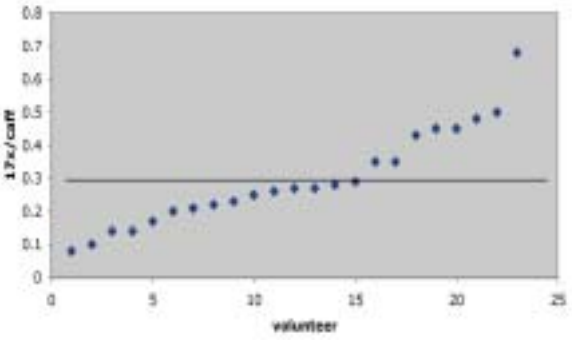


Fig. 1

Subject	17X/Caffeine	CYP1A2 Activity
1	-0.14	slow
2	0.08	slow
3	0.1	slow
4	0.14	slow
5	0.17	slow
6	0.2	slow
7	0.21	slow
8	0.22	slow
9	0.23	slow
10	0.25	slow
11	0.26	intermediate
12	0.27	intermediate
13	0.27	intermediate
14	0.28	intermediate
15	0.29	intermediate
16	0.35	fast
17	0.35	fast
18	0.43	fast
19	0.45	fast
20	0.45	fast
21	0.48	fast
22	0.5	fast
23	0.68	fast

Table 1

The Effect of Quantum Computation on Hash Functions

Emily Eder
Granada High School
Mentor
Terry Brugger
NAIC

Summary

Quantum computers will revolutionize computer science in the next 20 years. Conventional computer security procedures, currently suitable for classical computers, will become out of date. I show how the hash function, a central feature of message authentication, will be affected by quantum computers. Attacks on hash functions using classical computers are impractical because of space and time limitations. I developed a quantum algorithm that drastically reduces space and time requirements, both of which would make message authentication vulnerable to hackers with access to quantum computers. A central feature of the space reduction is the change from exponential to polynomial scaling.

The CH19 Form Designer

Abel Gezahegne
University of California, Davis
Mentor
Paul Amala
DCOM

ISCR Institute for Scientific Computing Research

Summary

The project was motivated by the need for biologists working on the chromosome 19 project to submit their findings either from experimental results or from other sources to a central database. A predefined interface they can use to submit results would either be too restrictive, too verbose, or even impossible in some cases because of the ad hoc nature of the experiments as well as the fact not all experiments can be predefined. Therefore, the objective of this project was to design and implement an application that the biologists may use to create their own interface tailored to fit any of the particular experiments they are performing. Using this custom interface, they can submit only the necessary information—nothing less, nothing more.

Under the direction of my supervisor, Tim Harsch, and two mentors, Amber Marsh and Julie Pitt, we designed and implemented an application in Java Swing that would allow the user to design a custom interface (form), use it to submit results to the database, as well as view previous submissions. To design a form, the user has a selection of predefined items that would be used to create an interface, for example, dropdown menus, multiselect lists, file selectors, text areas, and several others. The user will define the attribute of each of these items that make up the interface as well as the fields within the items. Once the interface is complete, the user has the option to save it as a template for future use. The user can then fill data in the designed form and submit the data, provided all the items have been filled with appropriate data. For each submission, the user is given a submission identification number that can be used for various purposes, including to view each submission.

As with any other application, modifications and improvements will be made to the CH19 Form Designer in the future. Aside from user-demanded improvements, some needs such as adding a date picker for creating a form, are already apparent for future versions. Functionality can also be improved so that users can not only view previous submissions but also update them.

Tools for a Statistical Package

Darryl Griffin-Simmons
Purdue University
Mentors
Dean Williams
&
Charles Doutriaux
EEBI

Summary

We seek to address the challenges of developing knowledge from global Earth system models by using statistics tools for analysis. Statistics tools are used in virtually every scientific project for analyzing and better understanding scientific data. The seamless integration of the Climate Data Analysis Tool (CDAT) and the Salstat statistical package provides climate scientists with a user-friendly tool for model diagnosis.

While at LLNL, I learned how to use CDAT software infrastructure subsystems, including:

- climate data management systems,
- a long masked numerical array operation called masked array, and
- Visualization and Control System.

I also modified the Salstat statistical package and worked with the CDAT graphical user interface (VCDAT).

Using SAMRAI and PETSc with the Immersed Boundary Method

Boyce Griffith
Courant Institute of Mathematical Sciences, New York University

Mentor

Richard Hornung
CASC

Summary

The immersed boundary (IB) method provides a mathematical and computational framework for addressing problems involving fluid–structure interaction and has proved to be especially useful in simulating biological fluid dynamics. To improve the efficiency of highly resolved IB computations, we have been developing an adaptive, parallel version of the IB method using the SAMRAI framework. Additionally, to overcome severe timestep restrictions found when using semiimplicit and explicit timestepping with the IB method, we are using PETSc to develop new implicit timestepping schemes. These methods should have the additional benefit of reducing splitting errors.

The work performed this summer is a continuation of work that began last summer as a Department of Energy Computational Science Graduate Fellowship laboratory practicum project.

The IB method describes the interaction of a viscous, incompressible fluid and an elastic structure. The approach used in the IB method avoids the necessity of remeshing the computational domain each timestep. It does so by describing the fluid on a Cartesian grid and the structure on a curvilinear mesh that moves with the fluid. A smoothed approximation to the Dirac delta function is used to connect these two computational domains. Using this smoothed delta function, quantities such as velocity may be interpolated from the Cartesian grid to the curvilinear mesh, and densities or distributions such as force may be spread from the curvilinear mesh to the Cartesian grid.

This summer, we have successfully implemented an adaptive version of the IB method using SAMRAI. Additionally, we have set up most of the data structures needed to interface this SAMRAI-based IB code with PETSc. We hope that we can take advantage of PETSc's facility for solving systems of nonlinear equations to implement fully implicit timestepping.

We intend to use this software with the 3D fluid-mechanical heart model of Peskin and McQueen. Additionally, we are interested in using the IB framework for an electrophysiological heart model with realistic (complex) geometry.

We are also interested in fully implicit timestepping. The PETSc interface developed this summer should facilitate this work.

Block-Matching for Video Object Tracking

Aglika Gyaourova
University of Nevada, Reno
Mentor
Chandrika Kamath
CASC

Summary

The growth of computer memory and processor speed has caused the expansion of automated systems for everyday applications. Computer vision systems constitute one example of such intensive computations systems, whose use becomes more feasible with the advances in computing power. Computer vision targets processing and analyzing visual information, which in general is characterized by large size and complex structure. The development of computer vision is hindered by the lack of complete understanding of how human beings process visual information. However, computer vision systems have already been incorporated in many practical applications (e.g., surveillance systems, medical imaging, robot navigation, and identity verification systems).

Object tracking is a key computer vision topic, which aims at detecting the position of a moving object from a video sequence. Models that describe road traffic patterns can be helpful in detection or prevention of uncommon and dangerous situations. Such models can be built by the use of motion detection algorithms applied on video data. Block-matching is a standard technique for encoding motion in video compression algorithms. It aims at detecting motion between two images viewed as a block. Each block of pixels in the current frame is matched to the best corresponding block of pixels in the destination frame.

We explored the capabilities of the block-matching algorithm applied to object tracking. The goal of our experiments is twofold: to explore the abilities of the block-matching algorithm on low-resolution and low-frame-rate video and to improve the motion detection performance by the use of different search techniques during the process of block matching. Our assumption for the process of video capturing is a motionless airborne camera. Although our experiments were made on tracking road vehicles (e.g., cars, trucks, motorcycles, and bicycles), the algorithm is general enough to be applied for tracking people or other moving objects.

Our experiments proved that the block-matching algorithm can be successfully used for object-tracking purposes from low-resolution and low-frame-rate video data. We observed that different searching methods have only a small effect on the final results and the additional computational load is not great. In addition, we proposed a technique based on frame history, which successfully overcame false motion caused by small camera movements.

Enhancements in VisIt

Akira Haddox
University of the Pacific
Mentor
Hank Childs
DCOM

Summary

VisIt is a parallel visualization and graphical analysis tool for scientific data used by many projects at Lawrence Livermore National Laboratory. VisIt is being actively developed, with new abilities and enhancements being requested and added. VisIt also has its share of reported bugs that need fixing. Readers need to be written for VisIt to support new users who use their own data formats. To meet other users' needs, various enhancements and readers must be coded, and bugs need to be fixed.

I implemented readers to support formats for Cosmos (LLNL's A Division), Mili (LLNL's B Division), and Boxlib (LLNL's A Division and Lawrence Berkeley National Laboratory). A new ThreeSlice operator was added that efficiently takes three orthogonal slices, aligned to the *x*, *y*, and *z* axes. I added another operator that removes cells from a dataset. A point tool was added to add interaction for the ThreeSlice operator, Streamline plot, and future point-based interactions. I contributed to speed enhancements in areas of mesh construction and contouring. I added several enhancements in the two-dimensional zooming interaction. Bugs were fixed in the KullLite reader, zooming interactors, mesh metrics, and the Box operator. To support the new Mili file reader, I implemented a parallel preprocessing tool to generate the metadata that VisIt needs, as well as added to VisIt an unstructured domain boundary exchange structure to create boundary information for datasets in serial or parallel.

VisIt is currently enhancing its AMR support, and when the support is completed, the Boxlib reader will be updated to reflect those changes to take full advantage of the AMR support. The unstructured domain boundary data structures will be expanded to exchange mixed materials and variables when a file format is added that requires that support.

Porting CALE to wxWindows

Jeffrey Hagelberg
Purdue University
Mentor
Paul Amala
DCOM

Summary

CALE (C Arbitrary Lagrangian–Eulerian) is a hydrodynamics simulation code written in C that was originally intended to run only on Unix and Linux systems. Over time, different versions have been created for various platforms such as the Macintosh. The primary goal of this project was to create a new version of CALE that would run on the Windows platform. Early in the project, it was decided that the best way to do this would be to use a platform-independent library so that we would be able to unify all of the various versions of CALE into one version that could run on all major platforms.

When we started, the first and most fundamental question we came upon was deciding what platform-independent package to use and what compiler to use. The compiler we chose was MinGW running under Windows 2000. MinGW is free and supports almost all of the standard UNIX header files. This universality was a huge advantage since it meant that most of the original CALE code did not have to be modified. The platform-independent package we decided to use was wxWindows. wxWindows was chosen because it is open source, robust, and provides support for drawing low-level graphics primitives such as lines and polygons in addition to providing support for creating dialogs, frames, and windows. Although it is primarily a graphics library, wxWindows also allows threads, files, and a variety of other things to be treated in a platform-independent manner. The new version of CALE was written in C++ because wxWindows was written in C++.

When implementing the wxWindows version of CALE, we had the command–prompt interface to CALE and the graphics window run in separate threads. This was challenging because the original version of CALE was single-threaded. We also wanted to keep the original CALE code and the new graphics code as separate as possible to minimize the coupling between the two. We achieved this separation by creating one class for each drawing event that occurred. For example, a class called FontSizeChanger was created that knows how to change the font size. All of these classes are subclasses of a base class called GraphicsEvent. All of the GraphicsEvent classes do not correspond directly to a low-level drawing call. In fact, the drawing they do could be arbitrarily complex. This distinction highlights one of the big advantages of this approach. It hides the details about how the drawing is actually accomplished and thus simplifies the task of implementing the drawing while making the drawing process less error-prone. An additional goal in creating the new version was to modernize the user interface. This came automatically when all of the dialogs were rewritten.

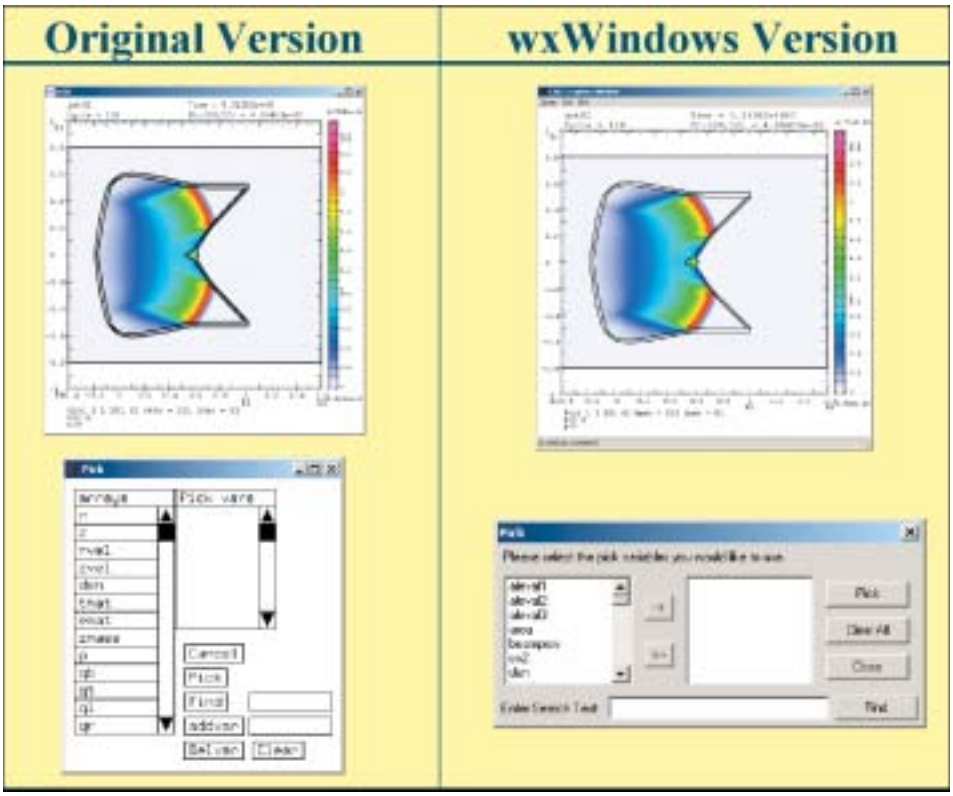
As of now, the wxWindows version of CALE only runs on Windows. Work is currently under way to make this version run on the Macintosh and Linux platforms as well. Additionally, more work needs to be done with regard to memory management and thread synchronization.

Continued

Jeffrey Hagelberg
Purdue University

Summary continued

The diagram below shows a comparison between some of the graphics features in the old version of CALE and those in the version using wxWindows. This example shows a typical CALE problem run.



Comparison of original version and wxWindows version of CALE

BlueGene/L: Providing Large Scalability for Scientific Computing

Amy Henning
University of California, Santa Cruz
Mentor
Don Dossa
CASC

Summary

BlueGene/L (BG/L) is a custom supercomputer providing massively parallel scalability across 65,536 compute nodes and 1024 I/O nodes with a target peak performance of 360 teraflop/s. Each compute node is composed of two system-on-a-chip PowerPC 440 processors with a nominal clock frequency of 700 MHz and an associated double pipe floating point unit, along with 512 MB of local memory. All compute nodes are interconnected by a three-dimensional torus, a global tree, and a barrier and interrupt network.

BlueGene/L is designed to deliver large-scale scientific computing to allow direct comparisons with real scientific experimentation and will result in faster runs than on previous supercomputing platforms. It is our goal to first run science applications and solvers and ultimately to incorporate techniques learned for scaling applications to 64,000 nodes to stockpile-stewardship codes. Efforts involved in developing BG/L will also present a promising architecture to be further scaled to the petaflop/s level.

For early software development purposes, IBM implemented a simulation environment for remote accessibility. The simulator (BGLsim) is a 64-way cluster running on a Linux microkernel. Efforts on both the simulator and real BG/L hardware involve porting, building, and evaluating scientific applications such as an optimized and "vanilla" version of the hydrodynamics code sPPM and a shock-wave solid-mechanics code, CTH, developed by Sandia National Laboratories.

Before we could begin porting to the simulator, it was necessary to implement a running version on a more stable and local platform such as the IBM SP and the Livermore Center for Advanced Scientific Computing's (CASC's) Linux cluster. BG/L uses the latest IBM XL compilers and the MPICH package, which, in turn, require us to use and build those packages on select platforms. For sPPM, we generated successful results on the CASC Linux cluster using gnu compilers and MPICH2 and on a Power3 SP (ASCI Frost) with XL compilers and MPICH2. On a smaller PowerPC cluster (ASCI Smurf), we were able to run a version of CTH. These applications could then be ported to BGLsim. It was essential to monitor the memory usage of the applications since BGLsim has a limitation of 256 MB per simulated node. Some modifications to source codes and compilation options were necessary for a temporary work-around. Working on an experimental simulation environment requires us to meticulously detect bugs associated with such features as the Fortran runtime and Makefiles hierarchies. Building large scientific applications on a variety of platforms presents a unique set of challenges and allows for systematic approaches to tackling bugs during compilation and runtime over time.

Work on the simulator resulted in successful runs of the vanilla version of sPPM with single precision using MPI and a single task run. We have a program provided by Bruce Curtis that verifies the correctness of output and has been thoroughly tested on Frost and Berg. It is our intent to use this checker when we obtain results from a double

Continued

Amy Henning
University of California, Santa Cruz

Summary continued

precision run of sPPM on BG/L. We are still in the development phases for CTH on BGLsim. Issues with the XL Fortran runtime and a few library routines need to be resolved by IBM before further progress.

Once we had a working version of an application running on the simulator, such as sPPM, we promptly ported it to actual BG/L hardware at IBM Watson Research Center in Yorktown Heights, New York. By running the application on the hardware, we can further evaluate its performance by obtaining timing measurements. Initial runs of an optimized version of sPPM show a 54% improvement in speed by simply using the XL compilers over GNU. Having employed only one hummer FPU at this stage of testing, we anticipate definite improvement in all applications with the utilization of double hummer FPU technology. With further runs to scale on the hardware, we will be able to formulate comparisons with other platforms and demonstrate the advantages of this architecture.

Continuous interests in BG/L led me to participate in research on job scheduler performance with Andy Yoo from CASC. Because of the massive number of compute nodes, having an efficient scheduler is essential. We are evaluating First-Come, First-Serve and Conservative-Backfilling algorithms in a small simulation environment evolved by using a bimodal hyper exponential distribution function as the basis for calculated results. The simulator generates performance measurements of the schedulers, including system use and response time. In order to conduct varied test cases, we made modifications to the simulator to allow for more parameter specifications. Such options include setting the degree of distribution functions or categorizing jobs by size or execution duration. We also added features to generate data files used for further visualization of our results. It is our prediction that the backfilling algorithm will provide minimal effects to system use and will not be efficient enough to implement on BG/L.

The ASCI BlueGene/L team has allowed me an opportunity to participate in countless meetings, telephone and video conferences, and visits to IBM Watson Research Center. Since BG/L is a collaborative effort between Lawrence Livermore and IBM, I attended a BG/L workshop in March where we were given a tutorial of the BGLsim and presented with information of progress by IBM on BG/L. Being part of the BG/L team, I attended the Conference on High-Speed Computing, where I had the opportunity to become acquainted with areas of research that ranged from memory bandwidth improvements to new hardware approaches to supercomputing. This conference gave me a broader prospective of what types of research I could possibly be involved in at the graduate level. At the conference, I was also able to present my work on BG/L by delivering a poster presentation and sharing what I've been involved in at LLNL.

In July, we had our annual review of the project. The review allowed a chance to interact with other collaborators and to get a more in-depth view of the overall efforts involved

Continued

Amy Henning
University of California, Santa Cruz

Summary continued

with BG/L. These past 6 months has given me the privilege to witness the intricate challenges and overwhelming rewards of creating a state-of-the-art supercomputer.

With Andy Yoo I am writing a paper entitled, “The Effectiveness of Backfilling on Improving Utilization of BlueGene/L,” anticipated in 2004.

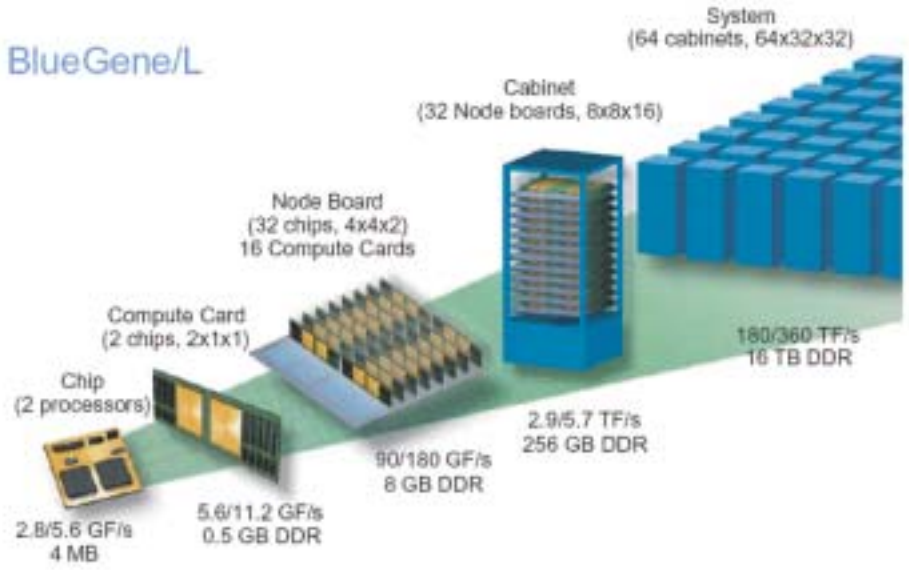


Fig. 1. The diagram of the BlueGene/L layout demonstrates the progressive development stages of the BlueGene/L hardware from a single application-specific integrated circuit chip to a full system.

Parallel Image Compositing Application Program Interface (PICA) Support for Chromium

Mike Houston
Stanford University
Mentor
Randy Frank
ICCD

Summary

The Parallel Image Compositing API (PICA) is designed to allow large parallel rendering applications to use a standard application program interface (API) for delivering content to various display sources using different imaging compositing systems. Chromium is a stream-processing framework for interactive rendering on clusters developed at Stanford University as part of a Department of Energy VIEWS grant. The objective was to add base PICA support to Chromium as well as provide an initial software compositing layer. Besides testing the feasibility of the current revision of PICA, this project sought to provide a compositing layer for use with VisIT and other LLNL visualization packages.

PICA support is now in the main Chromium CVS branch. PICA support is implemented as Chromium extensions (crPica), much as the parallel programming extensions such as semaphores and barriers are implemented. Several of the stream-processing units (SPUs) in Chromium required modification to correctly support the new API. Most just require dummy functions to be inserted to report errors, e.g., the crPICA calls really do not make sense in the “render” SPU. The “tilesort” SPU needed to be modified to correctly scale and position pixel data for large displays as well as correctly report the display size. The current, non-PICA compositing SPUs like “binaryswap” and “readback” can now correctly render to large tiled displays.

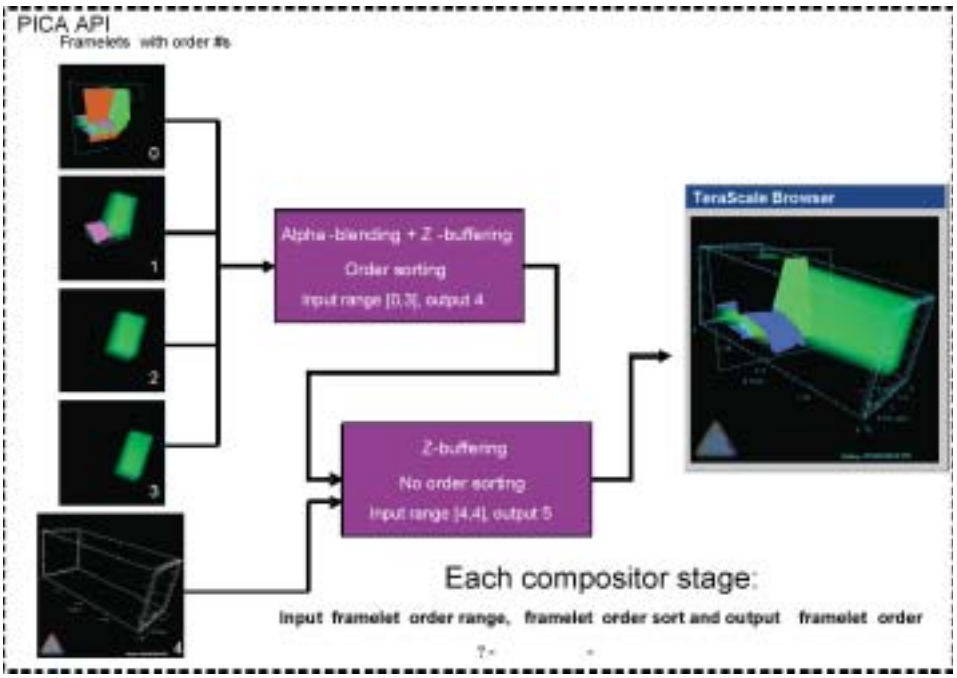
While implementing a software composting system for Chromium and PICA, a “composite” SPU, it became clear that the Chromium's networking layer had several major bugs. Most of the bugs where exposed because of the need to heavily use Chromium's out-of-band layer. High-speed network layers such as Myrinet and Quadrics were incorrectly using buffer pools, which were used to avoid expensive memory reallocation in DMA space. These bugs have been fixed as part of this effort, but several still remain, most notably the lack of support for nonblocking “connects” and “sends” in the most heavily used TCP/IP layer. Nonblocking communication is required for the compositing nodes to handle ordering problems during alpha compositing and to use more efficient compositing algorithms like “binaryswap” and SLIC.

I expect to finish fixing and optimizing Chromium's network layer by the end of the year. Besides generally improving the performance of Chromium, this work will allow me to finish a software composite layer as well as allow the design of SPUs that require nonblocking communication. Once hardware compositing systems designed for PICA become available from vendors, it will be interesting to see how they fit into Chromium and its PICA layer.

Continued

Mike Houston
Stanford University

Summary continued



A simple “composite” example using the TeraScale browser. Several nodes are handling the volume rendering, and one is handling the bounding box. The composite requires two stages, one stage for alpha blending the volume rendering and one stage for depth compositing the bounding box.

Volumetric Data Analysis and Compression

Lorenzo Ibarria
Georgia Tech
Mentor
Terence Critchlow
CASC

Summary

Scientific simulations produce large amounts of data, on the order of many gigabytes. With datasets this large, the storage space is costly, so compression must be addressed. Conventional compression techniques do not take advantage of the knowledge of the data or of its multidimensional coherence, and several compression methods that could use this knowledge are not ready for such a large amount of data. We have chosen to transform the data with a predictor and to apply a symbolic compression to the corrections. There are several kinds of predictors: interpolating, extrapolating, and fitting predictors. Our objective this summer has been to evaluate the effectiveness of each kind of predictor in the data, looking at factors such as the dimension of the predictor, its progressiveness, and how it performed against quantization.

Our tests were performed in the four-dimensional PPM simulation that represents two fluids mixing after a Mach 1 shock and in a series of two-dimensional images called the Miranda data, where two fluids of different density interact. Our results are not conclusive, and we advise to use a predictor of the highest order possible within the data features. The more quantized the data, using a simpler—and possibly an extrapolating—predictor is better.

Our last test was performed on a local n-dimensional extrapolating predictor, because it is useful in out-of-core compression and decompression. We studied its correction patterns and found that it had a much lower error bound along the axis. The ability to guess the performance of a predictor is a handy feature, because that information combined with a context arithmetic encoder can lead to great gains. Our naïve approach at using the prediction as a context resulted in a reported 25% increase in compression.

In the future, we plan to apply these ideas not only to large dataset compression and decompression, but also to their form of information, such as isosurfaces. We have seen that our predictors work well even outside a regular grid, and we are eager to follow leads into the isosurface time-dependent compression.

For more information on our predictor, see L. Ibarria, L. Lindstrom, X. Rossignac, and X. Szymczak, "Out-of-core compression of large n-dimensional scalar fields," *Proceedings of Eurographics 2003*.

Streaming Meshes

Martin Isenburg
University of North Carolina
Mentor
Peter Lindstrom
CASC

Summary

The standard indexed format for polygonal meshes is an array of *n* vertex records (e.g., *xyz*-coordinates, pressure values, surface normals, etc.) that is followed by *m* triangle records that index into the vertex array. When working with large meshes, this representation hampers efficient processing of the mesh data. In our work, we propose a novel mesh representation we call streaming meshes. In this representation, the vertex and triangle records appear interleaved, such that vertex records appear nearby the triangle records that use them. In addition, streaming meshes provide information about when vertices are no longer used (when a vertex is "finalized"). Large meshes in this representation can be efficiently streamed through limited memory, and seamless connectivity information can be made available along the active elements (those vertices—and their surrounding triangles—that have been introduced but have not yet been finalized).

In our work, we define streaming meshes, characterize their properties such as stream width (the maximal number of elements that are active at the same time) or the stream age (the longest time span that a particular element remains active). We show that certain processing methods already greatly benefit from a low stream width, while others also require the stream age to be low. In addition, we figured out how to create good streaming meshes from preexisting standard indexed meshes using an out-of-core method (a method that operates using only a limited amount of main memory, with the bulk of the data residing on disk). We also implemented an SM reader/SM writer application program interface (API) that allows mesh generation algorithms such as Marching Cubes to output a mesh directly in a streaming format. Finally, we designed a compression scheme that can compress streaming meshes on the fly in the order written through the API. The main challenge here, compared to previous work in mesh compression, was to account for the fact that the order in which the mesh is traversed is now determined by the process that writes the streaming mesh. Previous work in mesh compression always assumed that the entire mesh was available so that the compressor could choose its own traversal order, usually such that compression is maximal.

We have now a fully functional API for reading and writing streaming meshes. In addition, we have a mechanism that converts these streaming meshes on the fly into a processing sequence. In recent joint work with Stefan Gumhold and Jack Snoeyink, we have shown that processing sequences are useful for efficient out-of-core processing of gigantic mesh data sets. Now that we can efficiently generate processing sequences directly from the output of, for example, a Marching Cubes algorithm, we can streamline the entire mesh-processing pipeline. This ability will significantly speed up a previously cumbersome workflow. With streaming processing, we can likely rely solely on CPUs and no longer be bound to I/Os.

Babel: Now Serving Java

Sarah Knoop
University of Wisconsin, Madison
Mentor
Gary Kumfert
CASC

Summary

Our aim this summer was to enable Java on the server side Java of the language interoperability tool Babel. Previously, Babel facilitated communication among C, C++, Fortran 77, Fortran 90, and Python on both the client and server sides but the tool operated only on the client side of Java. With the work completed this summer, Babel is much closer to presenting a full set of services.

Babel connects to Java through the Java Native Interface (JNI), a Java platform feature that allows a Java code to interact/interoperate with the host environment of the machine (i.e., the operating system, native libraries, etc.). JNI is a two-way interface that Java programs to invoke native code (usually C or C++) and vice versa. For the complete server side Java functionality, the server-side bindings need to accomplish successful and type-safe conversions between the incoming Babel IOR (C-style) types and Java types. The converted Java types are the arguments that get passed to the Java server implementation. The reverse must be done with the return value from the Java code. It must be converted back to an appropriate Babel IOR type.

All Java server tests involving the above data types are building and passing successfully. We also have a good start on handling Object arguments; however, an unexpected error with the client side prevented full testing.

The following argument and return types have yet to be fully enabled and tested: Objects, Arrays, Enums, Opaques, Classes, and Interfaces. Also, a more succinct mechanism for server-side exception handling should be established and tested. Additionally, the management of local references in the server-wrapper code should be more carefully implemented. The generation code is designed so that much of the basic functionality needed by these additional tasks is already in place. Calling basic functions and implementing the details with respect to each task should be relatively easy to insert in the appropriate places, once they have been conceptually thought through.

Dual Least-Squares Methods for Computational Electromagnetics

Tzanio Kolev
Texas A&M University
Mentor
Panayot Vassilevski
CASC

Summary

The goal of this project was to develop efficient algorithms for the numerical solution of problems arising from various electromagnetic models. Specifically, we are interested in the system of Maxwell's equations and the related eigenvalue problem. These are important in practical applications such as the computation of the electromagnetic field generated by prescribed current and charges or in the computation of the eigenmodes that will propagate through a given medium. The work was carried out in collaboration with Joseph Pasciak from Texas A&M University and Panayot Vassilevski and Daniel White of CASC.

Our approach is based on very weak variational formulations of the div-curl systems corresponding to the electrostatic and magnetostatic problems. The finite dimensional approximation is a negative-norm finite-element least-squares algorithm that uses different solution and test spaces. This algorithm allows for approximation of problems with low regularity, where the solution is only in L^2 and the data resides in various dual spaces. The solution operators for the above problems are further used to obtain an approximation to the eigenvalue problem.

The resulting discretization method has the advantages of avoiding potentials and the use of Nedelec spaces. In fact, we allow for the mixing of continuous and discontinuous approximation spaces of varying polynomial degrees. Additional advantages are that the matrix of the discrete system is uniformly equivalent to the mass matrix and that spurious eigenmodes are completely avoided. Finally, the dual inner products can be efficiently implemented using preconditioner for standard second-order problems (for example, a sweep of multigrid).

A computer program was developed that applies the method to the Maxwell equations and the eigenvalue problem in the frequency domain. It is written in C++ in the framework of the AggieFEM finite element library, which supports complex geometries, local refinement, multigrid preconditioning, and OpenGL visualization. The code is based on the solvers for the magnetostatic and electrostatic problems. It works on triangular, tetrahedral, and hexahedral meshes. It provides an eigenvalue solver, which allows for computation of blocks of eigenvalues, and a solver for the full-time harmonic system. A parallel version of this code was implemented based on MPI and CASC's Hypre preconditioning library. It uses matrices built in parcsr parallel format, Lobpcg as a parallel eigensolver, and BoomerAMG or ParaSails as a preconditioner.

Various tests were performed to confirm the theoretical estimates for the least-squares method. Numerical experiments for the eigenvalue problem on a linear accelerator induction cell were compared with the those from the EMSolve project in CASC, which is based on Nedelec elements. The general observation is that the efficiency of the method on hierarchically refined meshes (where geometric multigrid can be used) can be preserved on unstructured problems if one uses algebraic multigrid.

Creating a Graphical User Interface for a Herbivore Population Simulation

Magdalena Kowalska
Warsaw University

Mentors

Tanya Vassilevska
&
Tina Carlsen
CASC

Summary

As part of a Department of Energy project to assess risk to herbivore populations from petroleum exploration, a simulation of a herbivore population was created for researching the effect of habitat reduction and fragmentation. The goal of the project is to develop an ecological framework for evaluating effects of environmental pollution on population persistence. The model for the simulation incorporates spatial movement and social behavior together with a large amount of quantitative information. Incorporating all of the data in the model is possible only by using a discrete individual-based model. The model is a nonlinear discrete time map, which transforms a certain number of matrices at time n into the same number of matrices in time $n+1$. Each individual animal is represented in this model as a vector of the characteristic attributes of its species. The population is a matrix consisting of such vectors. The habitat of the population is divided into square cells; consequently, the environment is represented in the model as a matrix of vectors of attributes of one cell. The discrete maps contain both deterministic and probabilistic rules.

The program that is based on this model has a complex input. It consists of several files containing environmental and species data used to generate the initial population of animals and the initial state of every cell. To evaluate the outcome of a discrete simulation containing probabilistic rules, one conducts a large number of simulation experiments that require frequent manipulation of the input files and the output format. To conduct these simulation experiments, the user must have a detailed knowledge of the code and the nature of its parameters. The code's intended users, most of whom are ecologists, do not have this detailed knowledge and thus find it difficult or impossible to work with the code. The solution to this problem lies in the construction of a graphical user interface (GUI).

We decided that a front-end architecture is most suitable for the project because it prevents interference with the structure of the simulation program. A GUI created in this way is a “wrapper” for the simulation. It allows for fast and convenient generation of input. The GUI incorporates the data from all input files used by the simulation, facilitates their adjustment, and passes them to the simulation in the form of a configuration file. We created a GUI that is appropriate for different simulations based on similar format of the input data. Thus the GUI has some features that allow for editing the format of data. In addition, users have an option to choose which simulation they want to run. The tool not only allows for the rapid creation of an application from scratch but also provides a number of templates to start with.

We plan a distribution of the software.

Investigation of the Oblique Shock Richtmyer–Meshkov Instability

Marco Latini
California Institute of Technology
Mentor
Oleg Schilling
Lawrence Livermore National Laboratory

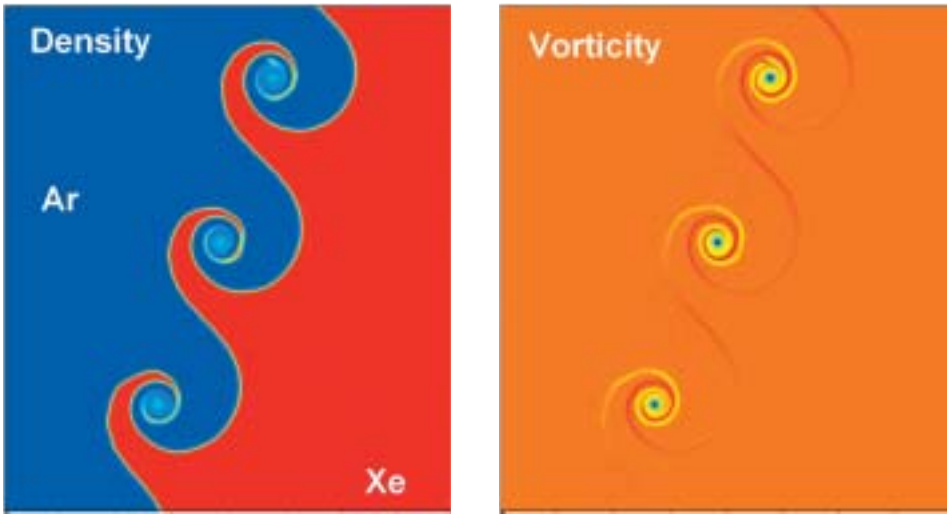
Summary

In the classical setup, the Richtmyer–Meshkov instability (RMI) is studied in the case of a normal shock interacting with a perturbed material interface. The shock deposits baroclinically generated vorticity on the interface, which drives the development of the instability and the formation of a mixing layer at the interface.

In the present work, we revisit the instability observed when an oblique shock interacts with a perturbed interface. We call this case the oblique shock Richtmyer–Meshkov instability. In our investigation, we use the weighted essentially nonoscillatory (WENO) high-order shock-capturing method and analysis to elucidate the complexity of the hydrodynamics resulting from the interplay of the Richtmyer–Meshkov and Kelvin–Helmholtz instabilities.

This study is motivated by current research and experiments on inertial confinement fusion (ICF) capsules. In an ICF capsule implosion, a laser drives a shock that compresses a deuterium–tritium (DT) mixture causing ignition and fusion. The mixing resulting from RMI and other instabilities limits the compression and the yield in energy. Oblique shocks may be caused by asymmetric drives. It is therefore important to study the hydrodynamics driving the oblique shock RMI.

In our study, numerical simulations were conducted using the WENO shock-capturing method developed at Brown University by Wai-Sun Don. This method provides unprecedented resolution of small-scale features through high-order reconstruction procedures. We present for the first time numerical simulations of the oblique shock



This figure represents density and vorticity plots for the oblique shock Richtmyer–Meshkov instability at time t = 2 ms after the passage of a Ma = 1.2 shock. The interface separates argon and xenon and was initially at an inclination of 75 degrees. The initial sinusoidal perturbation had an amplitude of a = 0.2 cm and k = 1.8888. The vorticity deposited by the shock drives the initial growth of the perturbation and the Kelvin–Helmholtz induced shear contributes to the formation of the roll ups. Numerical simulations are obtained using the WENO 5 method developed at Brown University.

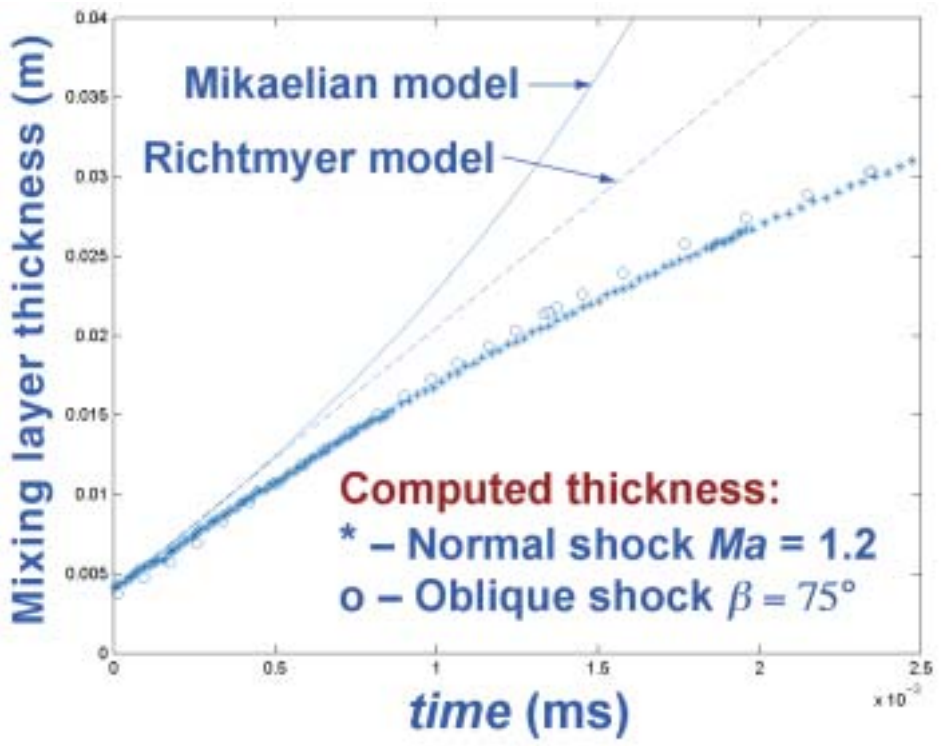
Continued

Marco Latini
California Institute of Technology

Summary continued

RMI, including a comparison to Mikaelian's prediction of the growth rate of the mixing layer. Our study suggests that Mikaelian's model captures the growth rate of the instability under limited circumstances, and we are currently working on improving the model. We identify the mechanisms driving the oblique shock RMI in the vorticity deposited baroclinically by the shock and the velocity shear as the shock passes. Furthermore, our results indicate that high-order reconstruction-evolution techniques provide conservative shock-capturing capability, as well as accurate resolution of small-scale flow features.

This investigation is one component of a larger program investigating physical and numerical aspects of complex hydrodynamics and turbulent mixing generated by accelerations.



In this figure we compare the growth rate of the mixing layer following refraction of a normal and oblique shock. In both instances the shock refracted at a speed of Ma = 1.2. We compare the computed growth rate of the normal shock case against the Richtmyer model and the oblique shock against the Mikaelian model. Our results suggest limited agreement with the Mikaelian model.

RETURN TO TOC

Creating a Standard, Extensible Mechanism for Native MPI Communication in Python MPI

Taylor Leese
University of San Francisco
Mentor
Pat Miller
CASC

Summary

Python MPI (pyMPI) is a Python interface to MPI and is designed to provide parallel operations for Python on distributed, parallel machines. There are certain difficulties in communicating data in pyMPI that are not present in C or Fortran. Python is an interpreted programming language while C and Fortran are compiled. Consequently, the data types of variables in Python are only available at execution time while in C and Fortran they are available at compile time. Since data types in Python are dynamically typed, it is necessary to compute these types during execution in order to ascertain the proper way of communicating them using MPI. Object serialization provides a solution for communicating dynamically typed variables since it isn't necessary to know the type of an object during serialization. However, the previous pyMPI design of using object serialization to communicate data in pyMPI creates unnecessary overhead. Removing the need to serialize all data in pyMPI was a major motivation in creating a new design for communicating basic Python types and Numeric Python (NumPY) arrays. Aside from the extra processing needed to serialize and de-serialize a Python object, the message sizes for serialized objects are much larger than the size of their respective Python data types. In turn, eliminating the need to use object serialization would improve the scalability of pyMPI and result in a design that creates a major improvement in overall performance.

To communicate data natively (without using serialization) using pyMPI, specific information about a particular Python data type must be known. Thus, a new design was created for pyMPI using a hash table (represented as a Python dictionary). Python objects are hashed using their Python type (PyTypeObject). Each entry in the hash table is a Python tuple containing three callable Python objects. Each callable Python object returns a Python tuple containing the information needed to either pack, unpack, or reduce the particular data type respectively. If a data type is not in the registry (the hash table represented as a Python dictionary), then the default is to communicate the data as a PyBaseObject_Type. Objects having PyBaseObject_Type or data types not in the registry default to using the cPickle module to communicate the data. Thus, the need for object serialization using cPickle has not been entirely removed from pyMPI, but instead, it is only used when absolutely necessary to communicate the data.

Initially, there are either four or five entries in the registry depending upon whether or not the NumPy package was installed on the system prior to compiling pyMPI. These types are PyFloat_Type, PyInt_Type, PyBaseObject_Type, PyString_Type, and possibly PyArray_Type if NumPy is installed. However, the system is not limited to these data types. Using a Python dictionary as a hash table allows the registry to be easily extended for new Python data types. Thus, the new design has created a standard and extensible mechanism for communicating Python data types without object serialization. It is straightforward for users to add their own Python data types to the

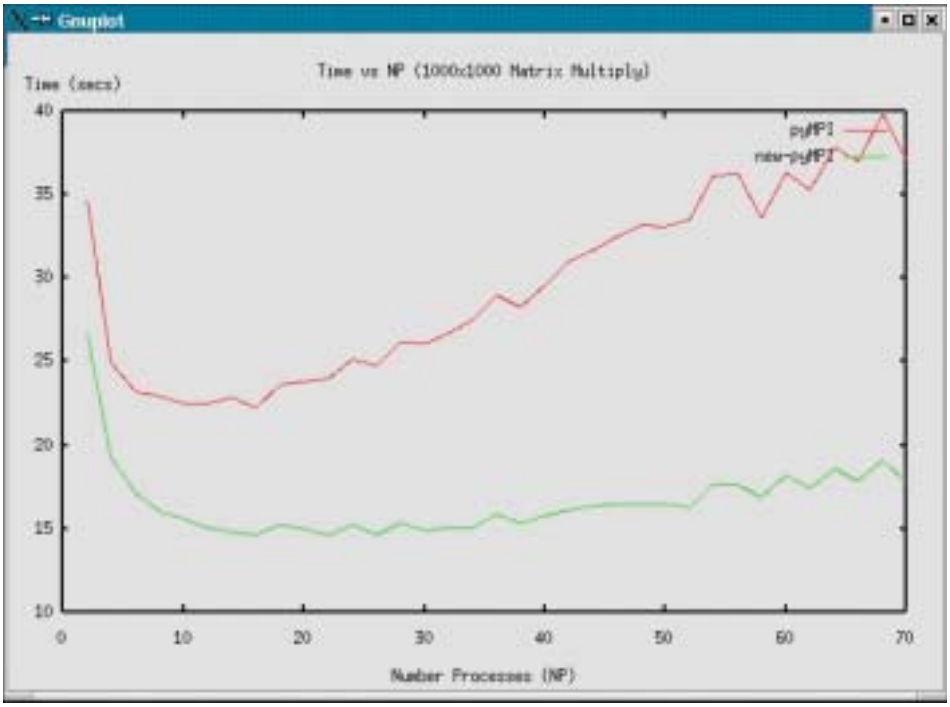
Continued

Taylor Leese
University of San Francisco

Summary continued

registry. The user must simply insert a tuple containing the three callable Python objects used to pack, unpack, and reduce the datatype into the pyMPI registry using the type of this new data type as the key. Currently, send, receive, sendrecv, broadcast, isend, and irecv all communicate data natively using this design.

Currently, there is no support for obtaining native communication in pyMPI for gather or scatter. This is an issue inherent to Python since the data being gathered or scattered is not necessarily composed of homogeneous types. Determining how to communicate nonhomogeneous data natively is an issue to be resolved at a later time. In the current version of pyMPI, this problem is handled by using object serialization since Python objects with nonhomogeneous data pose no problem for object serialization.



Time vs. number of processes for a 1000 × 1000 parallel matrix multiply. This graph shows the improvements in speed and scalability achieved by the new pyMPI design.

A Web-Based Genome Analysis Tool

Brian Lum
UC Berkeley
Mentor
Tim Harsch
EEBI

Summary

My main project was to design a web-based cross-genome analysis tool. Biologists use a tool that performs cross-genome analysis. Unfortunately, that program is built for a Unix system and is fairly complicated to use. I built a front end for them so they would have a user-friendly graphical interface.

Before I started on this project, I was given two other small projects that were mainly educational. These taught me about Perl DBI and CGI so that I could access databases and build a web interface. For my first project, I transformed a customer input into an XML format that was required by another analysis tool. The second project was editing a preexisting web interface to show more information from a database.

For my main project, I used the knowledge from the first two projects. I spent a great deal of time predicting all the ways user could give bad inputs and provided ways of handling those errors appropriately. The genome analysis tool takes a long time to run, so we want to check the input from users as much as possible beforehand, so we can catch the errors before using the program. After finishing that project, I made another analysis tool that performed the same operation, but for a different group.

Currently, the web-based interface does not give all the options of the tools. After talking to many biologists, we decided not to give those options for simplicity purposes. In the future, this might change. Also, there is currently a limited scope for which genomes are compared. We may decide to add different genomes.

Time-Varying Reeb Graphs

Ajith Mascarenhas
University of North Carolina, Chapel Hill
Mentor
Daniel Laney
CASC

Summary

Physical processes that are measured over time, or that are modeled and simulated on a computer, can produce large amounts of time-varying data that must be interpreted with the assistance of computational tools. A popular method is computing and visualizing isosurfaces of the data. Topological properties of the isosurface, such as the number of connected components, number of handles (genus), and number of voids, can be important aids in interpreting data. The Reeb graph is a useful topological structure to study these properties. Intuitively, the Reeb graph encodes topological changes to the isosurface as the isovalue is varied. In this project, we study the evolution of the Reeb graph of a time-varying function defined on S^3 × R and develop an algorithm to compute the time-varying Reeb graph for functions defined by piecewise linear interpolation from sampled data.

Research effort was largely directed toward development of the theory and algorithm to compute time-varying Reeb graphs. The result is a manuscript to be published at a future date. Implementation is in progress. This is the first work to address the study and computation of time-varying Reeb graphs. This research was in collaboration with Valerio Pascucci of Livermore's Center for Advanced Scientific Computing, Herbert Edelsbrunner and John Harer of Duke University, and Jack Snoeyink of the University of North Carolina at Chapel Hill.

Implementation and application to large time-varying data sets is in progress. The work will be carried out at the University of North Carolina.

Autonomous Motion Segmentation of Multiple Objects in Low-Resolution Video Using Variational Level Sets

Mark Moelich
University of California, Los Angeles
Mentor
Chandrika Kamath
CASC

Summary

My doctoral research at UCLA has focused on variational and statistical methods for tracking objects in video sequences. The group I work with at LLNL is primarily involved with data mining but has recently become interested in extracting and mining information from long, low-resolution (in both space and time) video sequences. The primary application is surveillance, and the emphasis on low-resolution was motivated by Livermore's experience handling large data sets. My objective for the summer was to develop an algorithm that is capable of isolating (segmenting) moving objects in low-resolution video sequences.

My approach was to use a region-based, variational level set method to isolate the moving objects. The algorithm that I developed is based on a functional (or energy) that depends upon the content of the current and preceding two frames of video. The functional is minimized to provide the desired segmentation by evolving a PDE in the level set framework. Although some loss in performance is expected at low resolution, this approach works well for the intended class of video sequences. In contrast to traditional image-processing techniques, our approach has a strong mathematical basis, and unlike local methods such as optical flow estimation, it is effective at low frame rates. This method also avoids some of the problems involved with using background subtraction.

There are several ways to extend the algorithm. The most important are to develop an algorithm that provides a frame-to-frame correlation of the objects and to develop the data-mining algorithms for which this algorithm was intended.

Adaptive Algebraic Multigrid and Frequency Filters

Arne Naegel
University of Heidelberg
Mentor
Rob Falgout
CASC

Summary

The need for large-scale simulation and the power of current and future parallel architectures make the development of state-of-the-art solution techniques an important task for today's research. A key question arising in this context is the solution of large linear systems that arise when problems formulated in terms of (systems of) partial differential equations are discretized.

By the algebraic nature of the problem, this question is closely linked to the spectrum of the matrices. The class of multigrid or multilevel methods incorporates the idea of treating all frequencies of this spectrum equally and therefore has optimal or near-optimal efficiency and convergence properties. For simple model problems with underlying geometric information, the original geometric multigrid has proved to be highly effective. Because these methods are somewhat limited when applied to large unstructured problems, algebraic multigrid (AMG) was developed, generalizing the basic idea, but without the need to rely on any geometric information. Briefly described, the key idea of the method we investigated and of this approach in general is to decompose the space into two parts. The first subspace is one in which a basic solver, the so-called smoother, can effectively be applied; it can typically be described by eigenvectors associated with large eigenvalues. Its complementary space can consequently consist of eigenvectors associated with small eigenvalues, and we construct solvers for this space using one or several of these algebraically smooth vectors as representatives or test vectors to construct an efficient method.

The UG software toolbox was used to implement an adaptive AMG version that is able to identify the space where the smoother is not efficient, to generate an appropriate test vector, and finally to improve itself using this information. Its basic principles were derived from a work by Ch. Wagner, whose method has been generalized to be applied to poorly scaled matrices. A variety of numerical tests for representative problems were performed to compare the original method and its modification. In the cases where the original method fails, convergence was established or convergence factors were improved by one order of magnitude. We additionally examined the behavior of the new method in possible frameworks for self-correcting schemes.

Though already responsible for some modest success, the method promises an even bigger potential and should thus be considered a work in progress. Through our analysis, we were able to identify strengths and weaknesses of the adaptive AMG approach taken and gained valuable insight into the crucial role of the test vector. Research will be continued in Heidelberg, and future work will be done in continuing collaboration with Robert Falgout and the Scalable Linear Solvers Group at Livermore.

3D Morse-Smale Complexes

Vijay Natarajan
Duke University
Mentor
Daniel Laney
CASC

Summary

Morse functions are used in differential topology to study the topology of manifolds. We use the results from Morse theory to study the topological features of natural phenomena. For example, in x-ray crystallography, we can reconstruct the geometry by following ridges connecting electron density maxima. A Morse–Smale complex partitions the 3D space into cells with similar gradient flow characteristics. In earlier work, we gave a combinatorial algorithm to construct the Morse–Smale complex for piecewise linear domains. In this project, we implemented the algorithm for tetrahedral meshes and also designed and implemented a visualization tool for looking at these structures.

We implemented the Morse-Smale algorithm to construct the descending 1-manifolds. We also designed and implemented a tool to help visualize these descending manifolds. The visualization tool was developed using C++ and VTK. It provides the user with the ability to visualize the input dataset as a mesh and as isosurfaces for different isovalues. The user can look at the entire set of descending 1-manifolds and filter them based on the function values. The user can optionally use a color map to code the function value on top of each of the above structures. We hope that this tool will help users to identify features in their datasets. It could be used as an exploratory tool.

A lot of additions can be made to the visualization tool. Looking at other substructures of the Morse–Smale complex might be useful for exploring the input dataset. We are working on computing and visualizing the ascending manifolds.

A Computational Design Tool for Microdevices and Components Used in Pathogen Detection Systems

Andrew Nonaka
University of California, Davis
Mentor
David Trebotich
CASC

Summary

Because of recent events including 9/11, the war on terrorism, and the war in Iraq, Lawrence Livermore has taken an increasing interest in biological pathogen detection. My summer research involves creating computational models for MEMS (microelectromechanical systems) devices for pathogen detection systems. A systemwide computational model will save time and money for LLNL scientists because they will have the ability to accurately predict if their devices will operate correctly prior to fabrication.

My objective for the summer is to become familiar with the software infrastructure currently in development to solve similar problems and to adapt the software to deal with biological flow through microchannels driven by a magnetohydrodynamic (MHD) pump.

The project will rely on a software infrastructure known as Chombo, which is designed to solve partial differential equations using finite difference methods on grids. The infrastructure EBChombo (embedded boundary) is an advanced version of Chombo equipped to handle the complex geometries found in MEMS devices. Both make use of adaptive mesh refinement (AMR) to determine which spatial locations in the problem require the most computations to solve accurately.

I am working with David Trebotich from LLNL and Greg Miller and Thomas Marshall from the University of California at Davis to reprogram the infrastructure to solve our problem. We are using a series of discrete hyperbolic and elliptic solvers to implement viscoelastic flow with MHD in Chombo. We have also learned how to program complex geometries using the EBChombo framework and have run two-dimensional examples of ideal polytropic gas dynamics using our geometry.

In the next few months, we plan on implementing a particle scheme to simulate proteins and other large molecules in the fluid. Then we wish to extend our algorithm for viscoelastic flow with MHD to EBChombo to handle more complex geometries. We also wish to implement the AMR algorithm to speed up our code. We will be able to verify our results with actual experiments being performed at UC Berkeley.

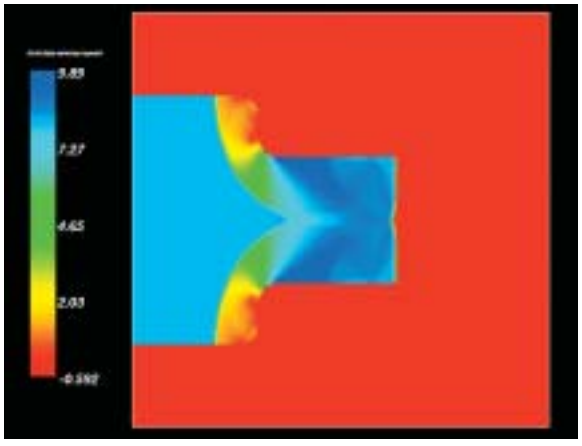


Fig. 1. Chombo output for an ideal polytropic gas. A shockwave of high-density and high-velocity polytropic gas has entered a contracting channel filled with low-density, stationary polytropic gas. Color scheme details the relative values of the x-component of fluid velocity.

NIF Automatic Alignment Archiving

Brian Overstreet
University of Colorado
Mentor
Punita Sinha
NIFE

Summary

The automatic alignment (AA) subsystem of Livermore's National Ignition Facility (NIF) laser system produces an enormous amount of data that must be analyzed by researchers to maintain system-critical components and learn about shots. The data must be archived to a database for easy analysis and not relegated to cryptic text files. The archiving mechanism developed to meet this need employs the languages Ada, CORBA, and Java to send the data to a database.

The AA control system was written in Ada, but Ada cannot communicate with the database because the database no longer supports it. Therefore, the decision was made to use Java's JDBC. The problem of interlanguage communication between Ada and Java came up, and the solution was to use CORBA's IDL. Another key issue was understanding the data. The AA system is large and complex and has many components. Once the data were understood, the next step was to draw an entity relationship diagram to design the database tables. With understanding of the data in place, the structure of the IDL was made to reflect the data. Then the Ada code was written to pull the data out of AA by putting it into the structure defined by the IDL. Finally, Java took hold of the CORBA. It unpacked any data type that the IDL sent it and repackaged it in an SQL statement to insert into the database. At the completion of this project, operators can archive the data with a single mouse click. Once the archiving process is complete, researchers can pull up a web tool to query the database and download the images associated with AA.

Future plans include writing a tool to link the multiple tables associated with automatic alignment into an easy-to-use interface.

Computational Gene Annotation and Human–Chicken Comparative Analysis

Samir Pandurangi
University of California, Davis
Mentors
Ivan Ovcharenko
CAR
&
Art Kobayashi
EEBI

Summary

This project involves a comparative study of the human and chicken genomes. The focus of the study will be on human chromosome 19 (Chr 19). This chromosome is unique in that it is relatively small yet contains a disproportionately large number of genes. Chromosome 19 in humans is closely associated with chromosome 28 (Chr 28) in chickens. The chicken organism turns out to be an ideal candidate for study because of its relative position in the evolutionary tree with respect to humans. Because of evolutionary conservation, the human and chicken genomes share many similarities.

Most of the chicken bacterial artificial chromosomes (BACs) in this study originate from chicken Chr 28. The first step of this study will involve annotating chicken genes—that is, using BLAST results to locate where genes possibly lie within the BACs. It is important to keep in mind that we are developing a technique for the prediction of coding structures in novel genomes. This method will make it possible to annotate any genome without a need for full RNA transcripts or heavy experimental evidence. These sequenced BACs homologous to Chr 19 comprise only a small portion of the chicken genome. However, if we are successful in annotating the genes that lie on these BACs, we can use our method to annotate any sequenced genome with very little experimental evidence. After annotation, we perform a comparative analysis of the chicken and human genomes.

The chicken genome has not yet been annotated. However, tools for gene analysis and annotation are plentiful. Some of these tools have been developed by organizations where researchers are conducting other annotation projects. One such tool, which is of central importance to this project, is the University of California at Santa Cruz (UCSC) genome browser. This is an online tool that helps to visualize genome characteristics. The browser that we installed locally was originally developed at UCSC (UCSC Human Genome Browser) to accommodate the needs of visualizing and annotating the human genome.

We have decided to use the framework of this tool to construct our own online browser to visualize the chicken genome. This involves a modification of the current tool provided by UCSC, an open-source C code. This modification is not trivial. The package involves tens of thousands of lines of source code with multiple, interdependent binaries and little documentation. I first had to understand the code, modify it to suit our needs, insert some of my own C code, and write other programs to interface with the existing binaries (see diagram).

After building the browser, I annotated the chicken BACs. Our browser currently has a wealth of information about protein sequence similarity to our chicken BACs through the many BLAST alignment hits. I use protein-similarity-driven gene predictors (e.g.,

Continued

Samir Pandurangi
University of California, Davis

Summary continued

GeneWise) to take these sections of similarity as input and output a set of predicted genes. In order to capture regions of the DNA that do not have any BLAST-produced protein similarity, I run other gene predictors (e.g., GenScan) that are indifferent about whether there is existing protein similarity in that region or not. This sheds some light on possible gene sequences between regions of protein similarity.

After completing annotation, my group and I will continue the project with a comparative analysis of the two genomes, using many of the same ideas and techniques developed in earlier parts of this project.

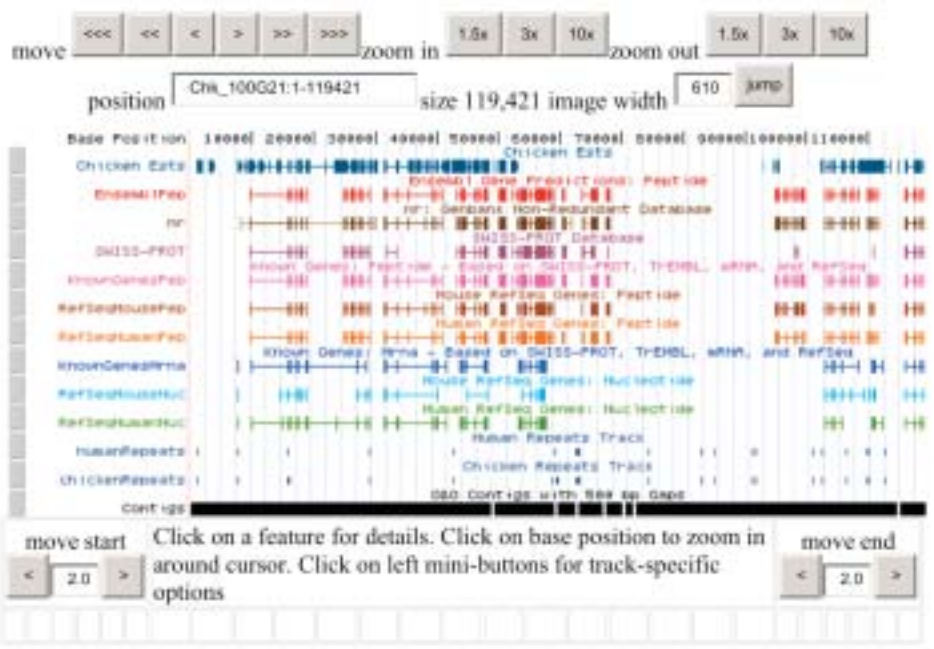


Fig. 1. UCSC Genome Browser on Chicken BACs

Memory Access Characterizations of Scientific Benchmarks

Ricardo Portillo
University of Texas, El Paso
Mentor
Jeffrey Vetter
CASC

Summary

The constant increase in raw processing power of contemporary chips has led to an improved performance on the computational side of a program's execution. And yet, the fact that performance in memory-access times and memory bandwidth has failed to keep up with this rate of improvement means that an application's overall performance is often hampered by bottlenecks in memory. To reduce these bottlenecks, efforts have been made to develop memory hierarchy systems that take advantage of memory-access patterns of applications. Further improvements can be achieved if memory systems are designed with access patterns of specific types of applications, such as scientific programs, in mind. To do this, metrics on how an application will access memory on a given hierarchy must be gathered in order to choose the most appropriate system for that application.

To gather these metrics, we have modified a version of the sim-outorder processor simulator, which is part of the Simplescalar suite of simulation tools. Sim-outorder is capable of simulating the out-of-order pipelined execution of an application running on a given architectural configuration, including the memory hierarchy, and produce a large range of performance statistics. Currently, sim-outorder supports the Alpha, PISA, and 32-bit PowerPC instruction set architectures. The most important modification to sim-outorder that we have made enables it to produce a detailed address trace of all the memory locations that an application accessed during execution and in the order that they were accessed. Using this data, we are now in a position to gather important metrics such as memory reuse distance and working set size of a program's execution that might enable computer architects to design complementary memory-hierarchy structures. In addition, these metrics may allow software developers to better optimize their code for existing memory hierarchies and therefore achieve better performance.

Because processor simulators such as sim-outorder may take hundreds or even thousands of times longer to execute than running the tested program natively, we have also added features that will allow us to simulate only small portions of a programs execution and thereby reduce the waiting time for simulation statistics. In addition, we have enabled sim-outorder to produce metrics on the instruction mix of a given program running on the simulated architecture. This information will give us more insight into what types of instructions are most used in a given execution and modify our simulated architecture configuration accordingly.

Computational techniques to extract meaningful metrics from the address trace produced by sim-outorder are now in development. Once these techniques are implemented, a study will be conducted on the memory access patterns of specific applications that are relevant to national security, including those which form part of the ASCI Purple suite of scientific benchmarks.

Out-of-Core Surface Parameterization and Remeshing

Serban D. Porumbescu
University of California, Davis
Mentor
Mark Duchaineau
CASC

Summary

With increased computing power, improved computational simulations, and improved sensor technology come vast amounts of information. To use this information, scientists must be able to visualize and manipulate this data in real time. Often, this data is too large to fit in the main memory of high-end workstations and, in some cases, even in the main memory of the supercomputers that generate the data.

Our continued research has focused on the development of a robust and efficient out-of-core surface remeshing and parameterization technique. These parameterized surfaces are amenable, among other things, to continuous-level-of-detail control, multiresolution editing, texture mapping, and, most importantly, compression.

Our technique begins with an arbitrarily triangulated surface. The end goal is to efficiently transform this surface into one that represents the same exact shape but is instead composed of very regular and similar-looking quadrilaterals (like those on a piece of graph paper).

Our past work has lead to an algorithm that constructs a hierarchy of surfaces that map to one another and that can be easily traversed. Our current work focuses on minimizing the distortion and maximizing the similarity of the quadrilaterals that comprise the surface.

We use a technique that is loosely based on the physical simulation of damped springs and masses coupled with the notion of a best-fit square (in the linear least-squares sense). The best-fit square represents what the perfect universe would look like to its respective quadrilateral if it were distortion free. These best-fit squares in combination with the spring simulation force distorted quadrilaterals to transform into similarly sized and shaped squares.

The algorithm is unique in that all operations require only local information. This translates into a technique that is easily made to work both out-of-core and in parallel.

This technique has been designed to work out-of-core, but the exact out-of-core algorithm needs further study and development. Our current distortion metric shows promise but needs further enhancement to ensure the resulting surfaces are manifold.

BlueGene/L Research Project

Daniel A. Reddeg, Jr.
United States Naval Academy
Mentor
Kim Yates
CAR

Summary

My research project at the United States Naval Academy focuses on supercomputing—its history, current uses, and where it is going in the future. My objective during this internship was to gain new insight into the world of supercomputing by learning as much as possible about the BlueGene/L project, as well as the many other supercomputing ventures taken on by LLNL. To this end, I sought to gain as much hands-on interaction with the different software and testing mechanisms employed in developing the new BlueGene/L system as possible.

My study of the BlueGene/L system was aided mainly by the BlueGene/L team, and their assistance in providing me with all of the materials for my research was greatly appreciated. In addition, I was able to test specific developments of the BlueGene/L system on the simulator, which provided me with raw performance data that, more than anything, verified the information I had received from the BlueGene/L team.

From the research I have done here, I plan to return to the Naval Academy and prepare a technical paper and presentation to submit to the research board at the Academy for review and approval of my final graduation requirements.

Data Visualization Project

Eric Scamman
University of California, Los Angeles
Mentor
Scott Brandon
DCOM

Summary

The purpose of this project was to assist Livermore physicist Scott Brandon in his validation and verification work with the Kull simulation code by providing visualization tools. The first tool needed was one to present multidimensional data in a visual format. By doing so, a large number of parameters could be compared at once to aid in error-limit estimation. The second tool's purpose was to display one-dimensional slices of data through the mesh used in a Kull simulation during and after a run. To be successful, it needed to be easily incorporated in the input deck for a Kull simulation.

The concept of the glyph was used to represent multidimensional data. Glyphs are small objects that can have many attributes and can be plotted over an *x–y* plane. Vectors in a vector field are examples of simple two-dimensional glyphs. Eventually, five-dimensional glyphs were employed, consisting of two vectors each and a color. Seven dimensions were successfully displayed with the plotter, which included options to filter parameters for certain value ranges along with other editing tools. The application was scripted in Python with the use of the PyGist, Tkinter, and Numeric modules.

The second tool was also constructed in Python using PyGist and Numeric. To perform a successful slice, the first objective was to build a line. This line consisted of a list of points separated by an increment of distance determined by the user. The essential function of the tool was to determine which zone each of these points lay in. The approach of breaking the mesh space into cubes allowed for fast searches over nearby cubes to find the point-containing zone. PyGist allowed a simple plotting medium to display the constructed line. Finally, code was added to the program to allow it to run over multiple processors in order to be used for very large simulation meshes. Upon completion, the code was successfully used to display plots of data vs. radius in the Sedov Blast Wave Test Problem over various user-defined lines.

The future of this project consists of the implementation of the above tools with Kull simulations along with improvements to the tools as necessary. Such improvements could include rewriting some Python routines in C and editing for more efficient scalability.

XS4C: Automatic Code Generation for Complex Step Method

Andrei Schaffer
University of Iowa
Mentor
Radu Serban
CASC

Summary

XS4C is an automatic code generation tool that parses a C user code and generates the C++ complex arithmetic version of that code. XS4C computes derivative information (gradients, Jacobians) via the Complex Step Method and is used in conjunction with LLNL's Sundials ODE/DAE numerical integration package. Although XS4C was initially designed for CVODE's integration package, which is part of the Sundials package, it is now a more generic tool for generating complex arithmetic C++ code. The complex arithmetic code that is obtained uses an enhanced version of the complex class provided by C++ Standard Template Library (STL). The complex code uses complex objects for scalars, and it duplicates arrays by providing a real-part array and an imaginary-part array, which are manipulated as blocks whenever possible for efficiency reasons.

XS4C parses the C source code using the C++ libraries provided by ROSE, a parsing tool developed at LLNL, and the complex arithmetic code is generated by attaching real and imaginary arithmetic code (expressions, declarations, function calls) attributes to each node in the Abstract Syntax Tree (AST), which was obtained as a result of the parsing operation. As a result, a set of C++ output files is obtained. These files are compiled and linked with Sundials libraries and with the user code, through a wrapper that is partially static and partially dynamically generated by XS4C. The result is an ODE/DAE numerical integration solver that obtains its necessary derivative information via Complex Step method by using the C++ routines generated by XS4C. Experiments performed so far show an increase in the accuracy of the solution compared to the finite difference computation of derivative information. This increase is because Complex Step method has the advantage over finite difference methods of avoiding catastrophic cancellations that usually accompany subtraction operations when step size is close to the round-off unit. As a result, Complex Step method can use step sizes as small as the round-off unit as opposed to step sizes that are typically proportional to the square root of the machine's round-off unit for finite difference methods—hence the improvement in the accuracy. It should be noted that the Complex Step method error is of order two when the step size is small enough. Therefore, if a step size proportional to the machine's round-off unit is used, the derivative information is as accurate as the analytic derivative computation performed on that machine.

The documentation and manual of XS4C are tasks that remain to be done. The second version of XS4C will provide a more efficient complex code by complexifying only a smaller set of variables and functions, namely only those that are in the intersection of program slice containing variables or functions that depend on the input variables (the independent variables, i.e., variables with respect to which derivatives are computed) with the program slice consisting of variables or functions that determine the output variables (the dependent variables, i.e., variables that are differentiated).

PFD*Now*: Desklog Module

Jonathan Schifman
University of Southern California
Mentors
Yousseff Abed
&
Priya Basu
EEBI

Summary

The duties of LLNL's Security Information Technology & Engineering (SITE) Group include developing information systems that enhance the security of the Laboratory and the efficiency of the Safeguards & Security Department. PFD*Now* is a web-based administrative system designed to automate, organize, and streamline the workflow processes that occur within the Protective Forces Division (PFD). The system involves several modules dealing with personnel, inventory management, officer scheduling, daily reporting, and workflow. The PFD*Now* system has been in development for over one year and is being used on a per-module basis.

Beginning in May 2003, the Desklog module became the highest priority item within the project. The Desklog module is designed to handle all information, workflow, and tracking regarding PFD incident reports, Central Alarm System (CAS) event logs, and shift logs for operations sergeants. By automating these tasks, all information is in a centralized location, which is accessible to any user with appropriate permissions. Desklog's main objective is to serve as a catalyst to daily PFD operational issues and track workflow in an easy to manage system.

The PFD*Now* application is a web-based Java application using a Model/View/Control architecture. All code was written using a framework authored by PFD*Now* architect Sean McFadden. The MVC framework uses JavaBeans, XML, and JDBC, interfacing with an Oracle 9i database, along with a single controller servlet, which handles server-side logic (model and control). The user interface (view) is presented using JSP pages. Priya Basu, Sean McFadden, and I met with PFD administrative personnel Clea Marples and Marlene Dutchover for most of the requirements analysis. All incident reports were previously paper forms, and event logs were completed using Microsoft Excel.

Each incident report was handled separately, but similarities between certain ones allowed us to design with an abundance of code-reuse techniques. The appropriate form fields were evaluated, as was the user interface and use case specifications for each form. Some reports facilitated special features, such as personnel assignment searches and calendar widgets for date and time selection. By mid-August, we were able to complete the automation of every major incident report in use by PFD.

The other main aspect of the Desklog involved the automation and integration of two disparate systems: the CAS Desklog and the Operations Sergeant Desklog. Much of the information recorded in these logs was redundant and caused unnecessary work to be done by both the operations sergeants and CAS operators within PFD. We were able to use a stripped down version of our base incident report form implementation to handle events. In the final system, the incident report Desklog and two event Desklogs have been unified in a single, easily manageable interface.

Full integration of email-based report distribution, notification, and reporting is currently in development. In addition, customized user alerts are the only other crucial component needed to begin user testing and training with the Desklog module of PFD*Now*.

MObIUS (Massive Object Integrated Universal Store)

Jennifer K. Sirp
California State University, Sacramento

Mentor

Terry Brugger
NAIC

Summary

General frameworks for distributed computing are slowly evolving out of Grid, Peer architecture, and Web Services. The following are results of a summer-long survey into distributing computing practices: First, Legion and Cactus-G have achieved the most in terms of providing an all-purpose application environment; second, extending a local programming environment to operate in a highly distributed fashion can be facilitated with toolkits such as Globus; and third, building a new system from the ground up could be realized, in part, by using some of the following components—an object-oriented database, Tapestry, JXTA, BOINC, Globus, component architecture technology, XML and related libraries, Condor-G, Proteus, and ParMETIS.

The objective of this research was to locate systems that could potentially contribute to the development of MObIUS, a general, distributed, object-oriented, framework. In contrast to current implementations of Web Services, Peer-to-Peer, and computational Grid middleware, the proposed system should be able to efficiently handle a wide variety of applications and adapt to a dynamic environment.

The approach taken was to first assess the state of distributed computing today. Next, systems both large and small were explored. Because time was short and it was unclear whether or not the proposed system existed, a broad approach was taken first. Later, the most relevant systems were reevaluated and questioned.

Distributed computing is an exciting topic; object-oriented technology and supercomputing are being reflected in this field of study. The whole concept of using the collective power of the Internet is fascinating. I am interested in seeing how advancements in component-based architecture, agent-oriented programming, and artificial intelligence will affect distributed computing in the future.

Data Visualization and Programmable Graphics Hardware

Jacob Stevens
Northern Arizona University
Mentor
Dave Bremer
ICCD

Summary

Recent innovations in hardware graphics technologies allow the traditionally fixed functionality of the graphics pipeline to be customized with short assembly language programs that are executed directly on the graphics processing unit (GPU). The objective of this study was to determine whether this new feature has the potential to benefit data-visualization techniques.

Several proof-of-concept programs were written that implemented key data-visualization techniques, such as paletted volume rendering, isocontouring, and simultaneous rendering of multiple volumes (see Figure 1). It was found that using programmable graphics hardware has the following notable benefits for data visualization, especially in the area of volume rendering:

- intensive per-fragment (pixel) computations are executed on the GPU, freeing the central processing unit (CPU) to perform other tasks,
- as much as 75% less video memory is required to render single-paletted volumes,
- fifty percent less video memory is required to render dual volumes,
- specialized GPU programs can be written to provide backward compatibility for older hardware operations that may be eliminated from future devices,
- when changing visualizations, only the currently active program on the GPU must be changed, rather than reprocessing an entire data set, thus allowing different visualization techniques to be explored interactively, and
- GPU programs have direct access to the real-time graphics application programming interface state, such as OpenGL (http://www.opengl.org), allowing visualizations that use techniques such as view-dependant lighting.

Overall, the use of programmable graphics hardware was encouraging, and although the technology has a tendency to be somewhat unstable and unpredictable because it was introduced relatively recently, it is likely to become an important tool in future visualization projects.

Because of the flexibility of using GPU programs to display data, an interface is being considered that will allow end users to have high-level language access to the functionality of programmable graphics devices. This will eliminate the need for assembly language programming and also provide the user with a rich set of useful mathematical operations commonly used in visualization.

Continued

Jacob Stevens
Northern Arizona University

Summary continued

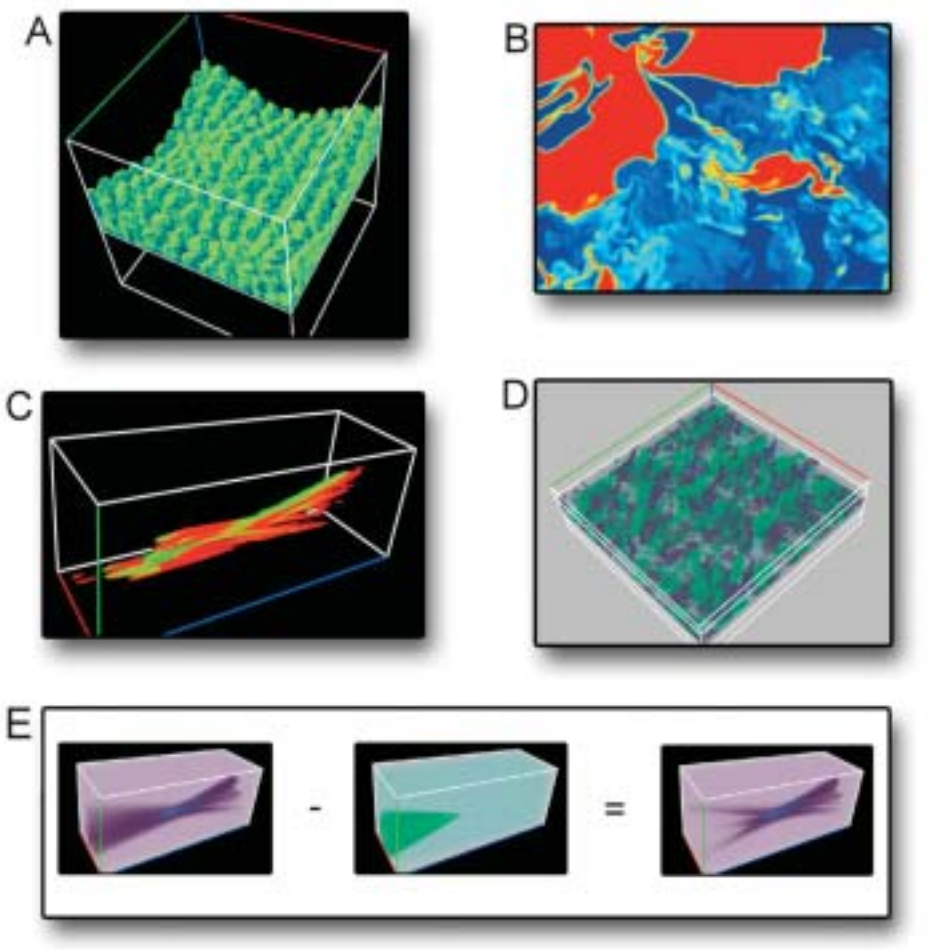


Fig. 1. Several volume renderings computed using programmable graphics hardware: (A) shaded volume rendering, (B) volume rendering with user-defined color palette, (C) gradient of data visualized as hue, (D) simultaneous display of multiple volumes, (E) point-wise subtraction of one volume from another.

Porting an Isosurface Calculation and Visualization Package from Yorick to Python, Utilizing SWIG

Mark Stuppy
University of Missouri, Rolla
Mentor
Steve Langer
DCOM

Summary

To increase the possible user base of an isosurface package written by Steve Langer, the decision was made to make it available in both Yorick and Python. These two scripting languages are functionally similar enough to allow the package to be used in basically the same way in both versions, but also different enough to give each version advantages and disadvantages. The goal of this summer project was to see if this addition could be done easily while adhering to set design goals and if so, to determine a method for doing so. These design goals are as follows:

- retain the same user-end functionality in the Python version as in the original,
- retain the same memory-management features in the Python version as in the original,
- complete the conversion with the least amount of scripting language specific changes to the original, compiled code as possible, and
- if possible, use the object-oriented nature of Python to extend the package's functionality through the use of object-member functions.

The project was first divided into two main goals: first, port the calculation functions into Python and second, port the visualization functions into Python.

For the first part, a code-integration package called SWIG was used to help wrap compiled C functions to be called from Python. SWIG's main job in this project was to automate the process of converting Python variables and classes into C-style variables and structures to be passed into compiled C functions. This process can be done by hand on a per-function basis, but in the past that has been shown to take a long time and to be quite prone to error. SWIG, however, only requires that developers write the upper-level code for converting object types or specific arguments, and then it generates the lower-level memory-management code and uses it wherever necessary. After wrapping some functions of this type, all that was left for the first part of the project was to mirror the structure and code of all of the associated Yorick functions in Python.

The second part of the project dealt with getting the same visualization functionality available in Yorick into the Python version. This required much more than simply wrapping more functions. It was also necessary to restructure some of the original code to be less dependent on certain Yorick-specific functions and therefore more interoperable with both languages.

The next step in this project is to add the multiprocessor calculation functionality created last summer by the author into the new Python version of code. This should be quick and painless; however, time constraints did not allow it to be completed this summer.

A Nodal Approach to Algebraic Multigrid (AMG) for Systems

Ryan S. Szypowski
University of California, San Diego
Mentor
Ulrike Yang
CASC

Summary

Algebraic multigrid (AMG) is a linear solver applicable to a wide range of problems. However, when the linear system to be solved is derived from a system of coupled partial differential equations (PDEs), AMG is often unsatisfactory for use as either a stand-alone solver or a preconditioner.

In such cases, it is necessary to augment the standard AMG scheme using knowledge of which unknowns are associated with a single node in the discretized PDE. One approach, known as the unknown approach, is to ignore connections between the different functions and coarsen them separately. Another approach, known as the nodal approach, is to categorize nodes of the discretization, as opposed to unknowns, as coarse or fine nodes, thereby keeping the coupled structure through all levels of the multigrid hierarchy. The nodal approach was studied and its performance as a preconditioner was compared to standard AMG and the unknown system version.

A large amount of code was added to a serial version of the Hypre library to accommodate block-structured sparse matrices. Also, special interpolation and relaxation routines were developed for use with the block sparse matrices.

When tested as a preconditioner for conjugate gradients on various two- and three-dimensional linear elasticity problems, the nodal AMG converged in fewer iterations than standard AMG. When compared to the unknown approach on simple problems, the nodal version was comparable, sometimes converging faster and other times slightly slower. However, on more difficult elasticity problems, the nodal version converged in far fewer iterations than the unknown version.

Although the results are promising, more theory and experimentation is required to derive better interpolation methods for nodal AMG.

Identity Access and Management Service

Robert Blake Taylor
Northern Arizona University
Mentor
Richard Mark
ICCD

Summary

The purpose of the project is to develop a new Identity Access and Management Service administration tool, replacing the current DCE tool, because of the deployment of a new security infrastructure for ASCI platforms and services within Livermore's Computation Directorate. The new administration tool must support a model that decouples the GUI from the back-end server, is platform independent, and has supporting libraries or components that allow for easy development within a distributed system. The prototyping of a new tool will also allow for the implementation of the necessary and requested changes within the current GUI design.

In developing the administration tool, our group met with the current DCE tool users to generate a list of the design changes to be implemented in the new GUI. The implementation of these design changes in the GUI has greatly simplified the process of completing what were previously complicated tasks. The GUI also implements the use of mnemonics and keyboard shortcuts, greatly increasing the speed at which users of the tool can complete the necessary tasks in updating user and group information.

Our group also focused on the task of designing the new system model that decouples the GUI from the back-end server. The new system model requires a way of sending data between the GUI and the back-end without tying the implementation to a single language. XML, which is language independent, was implemented as a means of storing the current state of this data for exchange between the GUI and the back-end.

The GUI currently relies upon the easy creation of serialized XML encoded user-defined structured data types, such as a User or Group, to update and change user and group information. Serialized types are not compatible between versions of the J2SE and therefore cannot be used in the final implementation of the GUI. The implementation of SOAP, a XML-based object-access protocol, is currently being researched as an alternative to using the serialized data types.

Implementing the SOAP protocol will allow the GUI, a SOAP client, to invoke procedure calls for the access and modification of user and group information on the back-end server via SOAP services located within the intermediate server. The implementation of a SOAP client and server will provide the flexibility required to allow the GUI or the back-end to be replaced or modified at any time.

Scientific Visualization Movies

Peter Tipton
University of Southern California
Mentor
Hank Childs
DCOM

Summary

After a physics simulation has been run and its output has been stored, the scientist who ran the simulation wants to see what happened. The process of actually looking at computer data from a physics simulation is called visualization.

Sometimes scientists want to see how the simulation progressed through time. They want to see a movie, which sequentially shows each time step of the simulation and displays whatever information is desired. This is a visualization movie.

Using the visualization tool VisIt, we can read in the data from a physics simulation and make a picture from it. We can show different aspects of the simulation, and we can show the simulation from different views. However, we can't make a movie in this manner.

To make a movie with VisIt, we have to run it from its command line interface. VisIt comes with the ability to make movies, but only with a scripting language. VisIt uses Python to do its scripting. By typing in various commands, we can tell VisIt what we want to see, where we want to see it, and when we want to see it. We now have control over the temporal aspect of a movie. We can change the view of the simulation throughout the movie, giving us a different perspective of things. We can show the outside of something and then reveal what is inside. In this manner, we can animate anything we want.

The other aspect of making a visualization movie is being able to show several smaller movies at once. We want to show one aspect of a simulation made with VisIt while simultaneously showing a different aspect of the same movie. We want to make a bigger movie that is a split screen of two smaller movies. This can be done with Unix Shell Scripting.

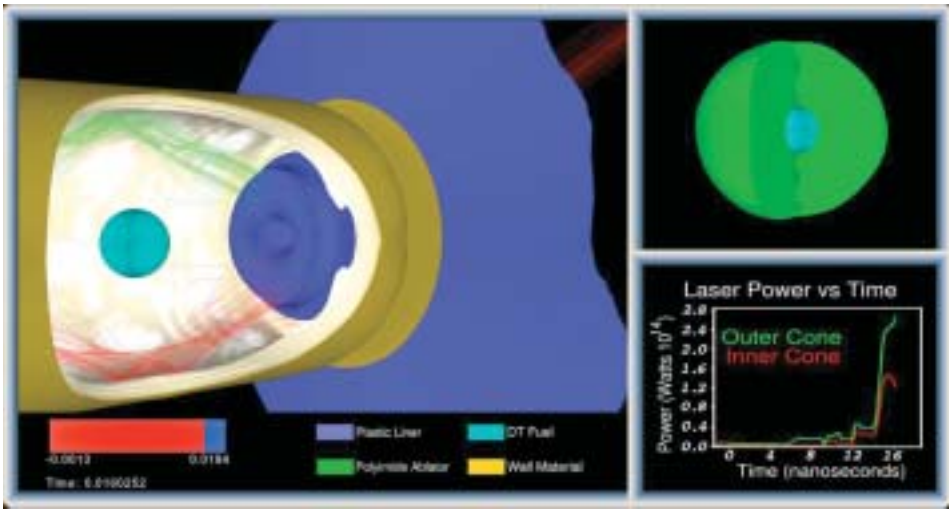
If we have two image sequences (two movies), we can issue various resizing and compositing commands from a Unix shell script that will go through all of the frames of the movie and composite them into one large movie.

This past summer, I used Python scripting along with Unix Shell Scripting to make movies in this manner. I have finished one movie for Michael Marinak, who is visualizing an indirect drive laser-fusion simulation of the National Ignition Facility (NIF), showing the ignition of the target. (On the following page is a picture from this NIF movie.) Having made this movie, I have learned a great deal about scripting, Unix, and parallel computing.

Continued

Peter Tipton
University of Southern California

Summary continued



Visualization of a simulation of laser-fusion ignition of an indirect-drive target on the National Ignition Facility.

Run Time Adaptive Load Balancing for Large-Scale Parallel Computing Systems

John Viles
Stanford University
Mentor
Jeffrey Vetter
CASC

Summary

The objective of my summer assignment was to evaluate the conceptual viability of a run-time load-balance control system for large-scale parallel applications. Effective load balancing is a difficult problem to solve statically and is especially difficult to manage in the face of variations in the run-time environment during the execution of a large job.

The goals for an effective run-time load-balancing scheme include:

- distributability and scalability with low-run-time complexity,
- adaptability to changes in the computing environment,
- monotonicity toward improved parallel program performance, and
- automatic implementation, requiring minimal programmer effort to effect.

Accordingly, the load-balancing control system developed in this work is designed to augment a general parallel program by providing adaptive self-balancing capabilities with a minimum of programmer or user intervention. The augmented executable code becomes capable of adaptively optimizing its own use of parallel resources in order to run as efficiently as possible and to facilitate maximum parallel scalability.

A working prototype of the control system was successfully implemented in Matlab and tested by simulation with several different algorithms for systems of up to 64 processors. In these trials, the control system was shown to be robust and effective at updating the computational load distribution in response to both normal variability (up to 10% random degradation) in underlying system performance characterized by effective processor speeds and communication bandwidth and latency and sudden, substantial changes (up to 500% degradation) in these parameters.

To be feasible, the load-balancing control system needs to be mathematically separable into one set of components that depend on the parallel algorithm to be optimized and another set of components dependent only on program input parameters and hardware performance attributes. The former can be generated statically at compile time, while the latter must wait for run-time instantiation.

The prototype developed this summer at LLNL automatically generates a separable performance model, consisting of one set of statically generated, machine-independent formulas that capture the time and communication complexity of the program, and a second set of dynamically evaluated architectural parameters and transformational operators. As measurements of processor and communication timings update the system parameters during program execution, the control system employs a parallel version of Newton's algorithm to reduce variations in parallel execution times across the processor domain. The result is a flexible system with low run-time demands on hardware resources.

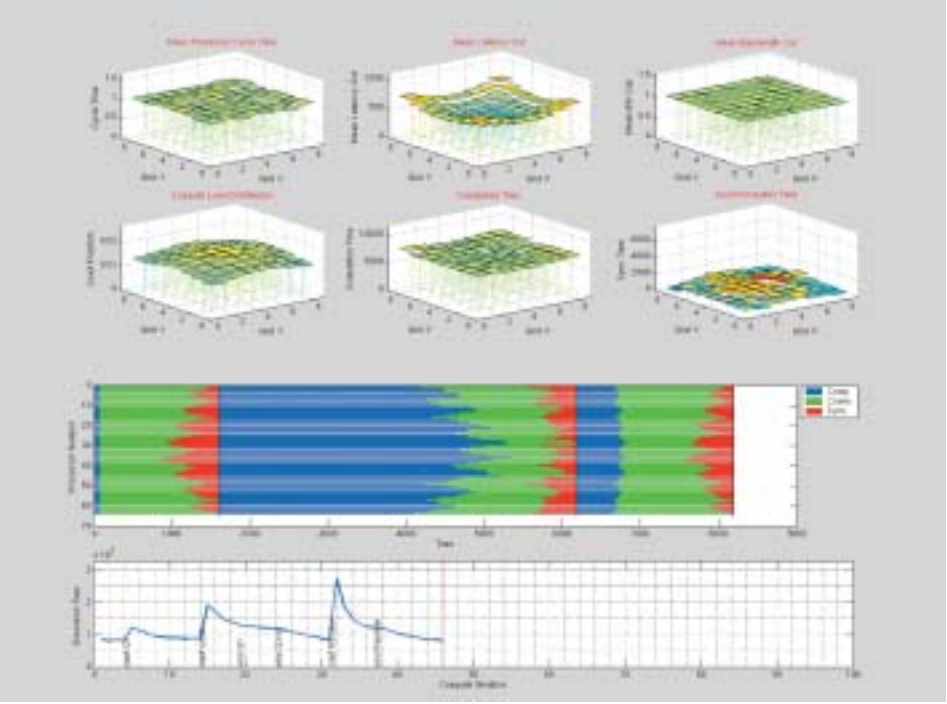
Continued

John Viles
Stanford University

Summary continued

Future work needed to support this project includes:

- instrumentation of MPI function calls at a low level to gather timing information without the intervention of the main user program,
- further analysis of the separable computational model to identify where it is applicable,
- implementation of a feedback control scheme to allow the control system to operate with a reduced fidelity performance model,
- extension of the performance model to manage functional parallelism effectively, and
- development of static program analysis tools to characterize run-time complexity based solely on either the source or intermediate representation of the program.



The effect of the load-balance control system on the execution time of a sequence of iterations of the conjugate gradient algorithm on a 8 × 8 (64 processors) grid. The hardware simulated includes a variation of 0–10% (“noise”) in communication latency and bandwidth and in processor throughput. The trace at the bottom of the figure illustrates the effect of three different major system disturbances on execution time and the recovery of the program as the load control system shifts computation to compensate.

Network Vulnerability Assessment Project

Qian Wang
Massachusetts Institute of Technology

Mentor

Terry Brugger
NAIC

Summary

The Network Vulnerability Assessment Project is the task of many computer security professionals under the Information Operations Assurance Center (IOAC) at LLNL. The project involves using tools, known as Deckard and MMM, to collect as much data as possible about a particular network. The tools create XML files of this data, which are read by the loader. The loader loads the data into the Network Mapping Database (NMDB) so the Graph Viewer can query and display the data in a user-friendly form. Computer security personnel can then use the Graph Viewer to look for vulnerabilities in the network.

Different computer security professionals work on the different aspects of the project. I helped improve the Deckard tool by creating a way to automate queries into the domain name database and to store the retrieved information. I also created two tools to monitor network traffic of an IP address and to detect the operating systems of machines based on packet information sent by such machines.

I learned the Python programming language to develop the DNS database text parser. Given a domain name, the parser automatically performs a Dig query. For each type of query, it reads in the lines produced by Dig and stores them into a file. Then it parses through the lines to determine whether the name provided is a valid domain name, if it is an actual domain name or just a single machine, the IP addresses of the domain, and the name and mail servers associated with the domain name. It stores each resource record into a list and makes sure there are no duplicates.

The fingerprinting tool uses the Disco fingerprint database to determine the operating system of a packet sender. However, it stores the fingerprint database information into a tree and then searches through the tree recursively to find matches. Disco searches through the database linearly and thus is not as time efficient as the fingerprinting tool. In each run of the tool, it creates the tree, listens for SYN/ACK packets or reads packet information from a file, compares the packet information to information in the tree, and then stores the possible operating systems of the sender into a vector. The fingerprinting tool ran about twice as fast as Disco. It can also find multiple possible matches for the OS, whereas Disco stops the first match it finds.

The tool can be vastly improved by incorporating some active fingerprinting techniques. NMAP searches through its active fingerprinting database linearly. That information could be stored in a tree to improve the speed of searching through the fingerprint database. I could use passive fingerprinting techniques to narrow the possible operating systems down and then only perform the active tests on the remaining systems. This would reduce the number of active tests needed and thus make this tool stealthier and more efficient than technology currently available.

Authentication Transport System

James Waslo
Northern Arizona University
Mentor
Terry Brugger
NAIC

Summary

The primary motivation for the creation of the Authentication Transport System (ATS) was to improve upon an existing authentication system. This system was necessary to support use of one-time passwords through the RSA SecurID system on nonsupported operating systems. The commercial system provided by RSA does not support all operating systems used by LLNL and the Computation Directorate. It was therefore necessary to create an interface between those systems and the RSA software. The existing system, DRAT, was fully functional, but with room for improvement. The DRAT system consisted of two essential components, a server and a client library. It used a custom encryption scheme to provide secure communications. The server ran in parallel with the RSA software. The client library allowed services on a nonsupported OS to use the OTP system. The goal of AST was to maintain this general structure, but create room for expandability and more reliable security.

To accomplish this goal, many of the elements of the software had to be redesigned. First, and most difficult, was the implementation of secure communications. It was decided to use OpenSSL, a commercially reliable, open-source SSL implementation. Use of this package required additional education because none of the participating programmers had any experience with SSL or OpenSSL. Several small programs, such as an echo client/server pair were written in SSL to familiarize the programmer with the OpenSSL library. Second, the server had to be designed and written to implement the desired feature set. Rather than completely create a new server, the existing DRAT server was modified. Much of the original initialization and setup of the server was kept. However, the core operations of the server were replaced entirely. This included changing the encryption to use OpenSSL and handling authentication requests. Authentication requests were handled by type, beginning with a standard OTP authentication and including the possibility of future varieties of authentication types. Depending on the authentication type, a subprogram called an Authentication Module was called on to handle the actual authentication. The overall effect was smaller, lighter, and faster authentication.

There still remain some things to be accomplished on the ATS. The client-side library needs to be completed. This library will be used in recompiling common services, such as FTP or SSH, to allow them to use the ATS. Also, certificate and key verification must be added to both the server and client library.

Newton-Krylov Methods for Expensive Nonlinear Function Evaluations

Rebecca Wasyk
Worcester Polytechnic Institute
Mentor
Carol Woodward
CASC

Summary

Newton–Krylov methods have proven useful for solving large-scale nonlinear systems. An advantage of these iterative methods is that they only require knowledge of how the Jacobian acts on a vector but do not require storage of the system Jacobian. A difference quotient evaluated at each linear iteration is often used to approximate this action without slowing the convergence rate of the method. For systems with expensive nonlinear function evaluations, however, the requirement of a function evaluation for each linear iteration can result in a costly computation. The goal of this project is to explore convergence rates and time savings associated with using different approximations to the system function in the difference quotient.

Theoretical results developed by Peter N. Brown, Homer Walker, and Carol Woodward indicate that making a linear approximation of the most expensive nonlinearities in the difference quotient would result in a method with the same convergence rates as those of the unmodified method, where the full nonlinear function must be evaluated in the difference quotient. Since making a linear approximation could be expensive, we also tested a method where the most expensive nonlinearities were lagged, so that they did not need to be recomputed in the difference quotient at each linear iteration.

These two approximations schemes were tested on several problems using the KINSOL and CVODE solvers in the SUNDIALS program, a suite of solvers developed at LLNL for solving nonlinear algebraic systems, ordinary differential equation systems, and differential-algebraic systems. For most of the problems tested, the linear-approximation method did converge in the same number of iterations as were required when using the unmodified difference quotient. A small amount of time was saved when using this linear approximation on a few problems, but none of the test problems had expensive nonlinearities where we would expect to see the most time savings. Although using the lagged values of the nonlinearities was not supported by theory, the increase in the number of iterations required to reach a solution was not great for most of the problems tested. So, even though the number of iterations increased, in general, when using this approximation, time was saved while solving a number of problems, and these savings tended to be more significant than those that resulted from using the linear approximation.

Future work aims to develop some theoretical results on solving time-dependent problems using these approximations in the difference quotient. More tests are also planned on problems with more expensive nonlinearities, where the modifications made could really be beneficial in terms of run-time savings.

Deriving Prolongation Operators for Algebraic Multigrid (AMG) Using Compatible Relaxation

Dominique Wiest
University of Washington
Mentor
Charles Tong
CASC

Summary

Iterative methods are an accepted way of solving systems of linear equations with many unknowns. As these systems grow in size with the advent of parallel computing, scalable methods such as algebraic multigrid (AMG) have become increasingly important. Recent advances such as compatible relaxation promise to expand the scope of problems AMG can reliably be used to solve. One such type is solving nonsymmetric linear systems. We have looked at deriving new prolongation operators for such systems based on the improved coarsening provided by compatible relaxation. We compare these new prolongations on several convection–diffusion problems.

The AMG has three distinct parts: smoothing, restriction, and interpolation (prolongation). For nonsymmetric operators, the relationships among these are more difficult to analyze. For a symmetric positive definite operator, restriction and interpolation can use the variational property, which is easy to implement and commonly used. Focusing on the interpolation operator, we decided to examine the performance of an interpolation operator based on the idea of compatible relaxation (CR).

CR is a modified relaxation scheme that keeps the coarse level variables invariant. Compatible relaxation is used to determine coarse grid variables independent of any geometric information from the physical system. The convergence rate of CR can be used as a general measure for the quality of the set of coarse grid variables. In short, CR is based on the assumption that error after smoothing is small on fine-grid variables.

Since compatible relaxation can be viewed as a measure of coarse-grid quality, it is worthwhile to explore using CR to form the interpolation operator. However, CR theory is based on symmetric positive-definite systems, and it has not been proven to be a good measure on nonsymmetric systems.

However, the convection–diffusion problems investigated this summer using a compatible relaxation-based interpolation operator outperformed the standard interpolation operator found in Ruge–Stuben AMG. In some cases, it enables convergence where AMG had previously failed.

These preliminary results suggest that compatible relaxation is a viable means to improve the interpolation operator for these types of convection-diffusion problems. Future work needs to be done to examine other kinds of convection-diffusion problems and also using different smoothers in the compatible relaxation process.